

Supporting Virtuosity and Flow in Computer Music

CHRIS NASH

December 2011

THIS DISSERTATION IS
SUBMITTED FOR THE
DEGREE OF DOCTOR
OF PHILOSOPHY



**UNIVERSITY OF
CAMBRIDGE**
Computer Laboratory



**ST JOHN'S COLLEGE
UNIVERSITY OF CAMBRIDGE
1511 - 2011**

Supporting Virtuosity and Flow in Computer Music

© 2011 Chris Nash. All rights reserved.

This dissertation is the result of my own work and includes nothing that is the outcome of work done in collaboration except where specifically indicated in the text.

No part of this dissertation has already been, or is currently being submitted by the author for any other degree or diploma or other qualification.

This dissertation does not exceed 60,000 words including tables and footnotes, but excluding appendices, bibliography, photographs, and diagrams.

All brand names and registered trademarks used in this work are the property of their respective owners.

Supporting Virtuosity and Flow in Computer Music

Chris Nash

As we begin to realise the sonic and expressive potential of the computer, HCI researchers face the challenge of designing rewarding and accessible user experiences that enable individuals to explore complex creative domains such as music.

In performance-based music systems such as sequencers, a disjunction exists between the musician's specialist skill with performance hardware and the generic usability techniques applied in the design of the software. The creative process is not only fragmented across multiple physical (and virtual) devices, but divided across creativity and productivity phases separated by the act of recording.

Integrating psychologies of expertise and intrinsic motivation, this thesis proposes a design shift from usability to virtuosity, using theories of "flow" (Csikszentmihalyi, 1996) and feedback "liveness" (Tanimoto, 1990) to identify factors that facilitate learning and creativity in digital notations and interfaces, leading to a set of design heuristics to support virtuosity in notation use. Using the *cognitive dimensions of notations* framework (Green, 1996), models of the creative user experience are developed, working towards a theoretical framework for HCI in music systems, and specifically computer-aided composition.

Extensive analytical methods are used to look at corollaries of virtuosity and flow in real-world computer music interaction, notably in *soundtracking*, a software-based composing environment offering a rapid edit-audition feedback cycle, enabled by the user's skill in manipulating the text-based notation (and program) through the computer keyboard. The interaction and development of more than 1,000 sequencer and tracker users was recorded over a period of 2 years, to investigate the nature and development of skill and technique, look for evidence of flow experiences, and establish the use and role of both visual and musical feedback in music software. Quantitative analyses of interaction data are supplemented with a detailed video study of a professional tracker composer, and a user survey that draws on psychometric methods to evaluate flow experiences in the use of digital music notations, such as sequencers and trackers.

Empirical findings broadly support the proposed design heuristics, and enable the development of further models of liveness and flow in notation use. Implications for UI design are discussed in the context of existing music systems, and supporting digitally-mediated creativity in other domains based on notation use.

Acknowledgements

Many people have helped and supported me during my work on this research. I'd especially like to thank my supervisor, Alan Blackwell, for his unwavering support and many insights, and from whose unfathomable depths (and breadths) of knowledge, I have learnt so much. Similar thanks go to Ian Cross, at the Centre for Music and Science, for his support in supervising me.

The whole Rainbow Group, along with my colleagues at the CMS and St. John's College have made my time in Cambridge a warm, friendly, and vibrant experience. Darren Edge, Daniel Bernhardt, Bjarki Holm, and Sam Aaron especially, have not only been invaluable sources of feedback and insight in my work, but much valued friends, inseparable by distance. Likewise for my college brethren, among them Matt Dolan and Moira Smith – who are always there when I need a pint or curry.

This work would never have been possible without the financial support of both Russell Ambrose and the Harold Hyam Wingate Foundation, and their valued enthusiasm for both music and inter-disciplinary research.

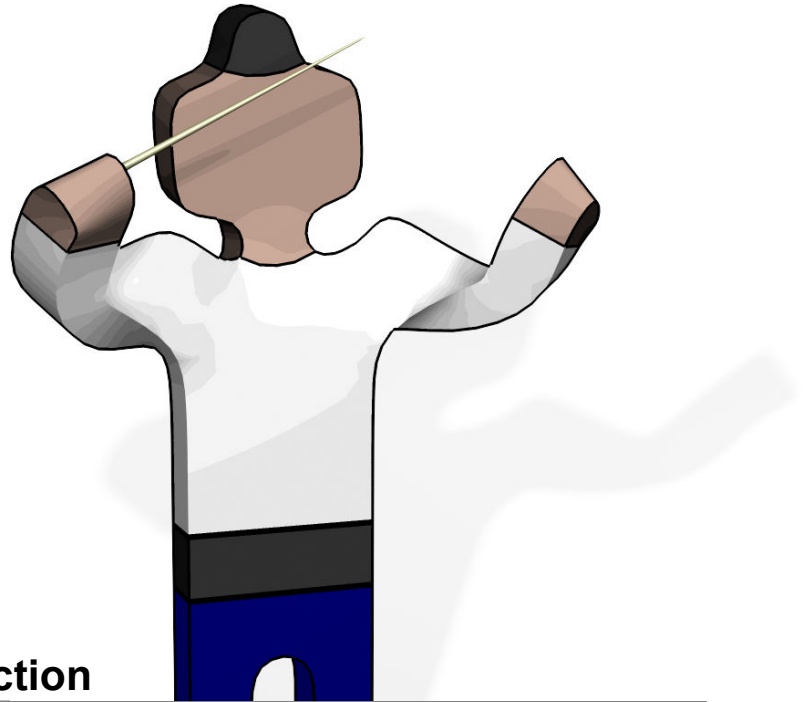
I am also grateful to everyone in the *reViSiT* user community, whose openness and enthusiasm has enabled this research, especially Maarten van Strien, Martin Fegerl, Hans Terpstra, and Esa Ruoho. I hope to make up for the lack of development during this research, with lots of new “scientifically-proven” updates!

Above all, however, I thank my family – Mum, Dad, and Kate – for their support, love, and patience. For a quiet, loving environment to live, grow up, and write up. But most of all, for inspiring my passion for music, and the drive to create and learn.

For my parents

Abstract	3
Acknowledgements	5
1 Introduction	9
1.1 Thesis Summary	11
1.2 Aims and Objectives	12
1.3 Research Contributions	12
1.4 Structure of Dissertation	12
2 Sequencing, Tracking, and the Demoscene	15
2.1 Digital Audio Workstations (DAWs)	16
2.2 Soundtrackers and the Demoscene	22
2.2.1 Technical Overview	23
2.2.2 Social Context	27
3 Creativity, Expertise and Motivation	31
3.1 Personal Creativity	33
3.2 The Creative Process	41
3.3 Creativity Synthesised	47
3.4 Supporting Creativity	49
3.5 Creativity in Music	56
3.6 Developing Musical Expertise	66
3.7 Motivation and Flow	74
4 Towards Digitally-mediated Creativity	84
4.1 Supporting Virtuosity in Computer Music	85
4.1.1 Design Heuristics for Virtuosity-enabled Systems	86
4.2 Systems of Musical Flow	97
4.2.1 From Virtuosity to Flow	97
4.2.2 Abstracting the Creative Music Process	100
4.2.3 Systems of Musical Flow	101
4.2.4 Modelling ‘Liveness’ in a Musical System	109
5 iMPULS: Internet Music Program User Logging System	115
5.1 Objectives	116
5.2 The <i>reViSiT</i> Soundtracker	117
5.2.1 Background	117
5.2.2 Program Overview	117
5.2.3 Reception and User Community	120
5.2.4 Development and Testing	122
5.2.5 Distribution	126
5.3 System Architecture	126
5.3.1 User Registration and Identification	127
5.3.2 Data Collection	128
5.3.3 Data Delivery	131
5.4 Interaction Visualisation Environment (IVE)	133
5.4.1 Visualisation and Analysis	135
5.5 Running the Experiment	140
5.5.1 Testing Experiment Code	140
5.5.2 Experiment Launch	141

6 Video Study: Tracking Composition Practices	143
6.1 General Observations	145
6.2 Chronological Overview	149
6.3 Evidence for Flow and Virtuosity	155
7 Keyboard Use and Motor Learning in Tracking	157
7.1 Speed and Timing	158
7.2 Keyboard and Program Knowledge	161
7.3 A Descriptive Model of Tracker Interaction	163
7.4 Developing Fluency in Soundtracking	165
8 Focus and Feedback in Digital Music	173
8.1 Activity Profiles	174
8.2 Measuring Liveness	184
8.3 Direct Manipulation for Audio-based Programs	188
8.4 Visual Feedback and Window Use	194
8.5 Case Study: <i>Image-Line FL Studio</i>	202
9 Flow and Cognitive Dimensions in Notation Use	207
9.1 Interaction Styles and Preferences	209
9.2 Flow and Experience	211
9.3 Cognitive Dimensions and Flow Profiles	213
9.4 Modelling Flow with Cognitive Dimensions	218
10 Conclusion	229
10.1 Summary of Findings	230
10.2 Methodological Review	234
10.3 Towards a Theoretical Framework	237
10.4 Future Directions	239
Bibliography	243
Appendix A	Tracker Effects Reference
Appendix B	Flow in Music Composition
Appendix C	<i>iMPULS</i> Questionnaires
Appendix D	<i>iMPULS</i> Interaction Event Types
Appendix E	<i>iMPULS IVE</i> Visualisations
Appendix F	<i>reViSiT</i> Software Updates
Appendix G	Overview of sampled users and data



chapter one introduction

... one can ask how deep a union research in musical controllers will be able to forge with the larger field of Human-Computer Interfaces, which generally emphasizes ease-of-use rather than improvement with long years of practice.

– Paradiso and O’Modhrain (2003)

As we begin to realise the sonic and expressive potential of the computer, *human-computer interaction (HCI)* researchers face the challenge of designing both rewarding and accessible user experiences that enable individuals to effectively control and explore complex creative domains such as music.

Exploratory creativity relies on expertise and intrinsic motivation, contrasting goal-based usability approaches in mainstream HCI practice and analysis. In music, creative individuals not only develop virtuosity with instruments, but also notations; and, while new performance devices provide new modes of realtime musical expression, relatively little research has looked at the composer’s use of notation, and how it can be supported by the computer, as a tool for sketching creative ideas.

Conventionally, both professional and amateur music production uses a mix of virtual and electronic devices integrated through a sequencer-based *digital audio workstation (DAW)*, which couples the capture of a live (realtime) performance with offline (non-realtime) *graphical user interface (GUI)*-based editing. In the process, the user’s creative process is not only distributed across

multiple devices, but also across distinct creativity and productivity phases, separated by the act of recording. An important question is how to bridge this divide between the musician's direct control and intimacy with hardware and the virtual environment of software.

This research looks at more pervasive use of digital notations in computer-based musical creativity, exploring digital approaches to editing notations that maintain a sense of “liveness” (Tanimoto, 1990) and immersion in the musical domain, while avoiding both the “indirect involvement” associated with overly formal or abstract notational layers (Leman, 2008) and the reliance on traditional musicianship. Drawing on creativity and musicology research, this thesis argues for a design shift from usability to virtuosity; proposing notation-based interfaces for composition that support the development of skill (motor and memory), and where a concise visual representation is closely-integrated with the high-availability of musical (sound) feedback, in order to maintain a musical context during offline editing.

At the same time, while developing musical virtuosity is historically associated with years of tuition and deliberate practice, the computer's capacity to respond to manipulations of a notation with rapid musical feedback can enable an intrinsically rewarding experiential learning process based on creativity, listening, and tinkering with music. Musical performances are scripted by the user, but executed by the computer, allowing interaction at a pace that is not only accessible to novices, but also increases the scope of what experts can express beyond what is possible in realtime.

Integrating psychologies of expertise and intrinsic motivation, flow theory (Csikszentmihalyi, 1996) is used as a framework for identifying and analysing the properties of a notation that support learning and creativity, and is combined with the *cognitive dimensions of notations* framework (Green, 1996) to develop models of the creative user experience, working towards a theoretical framework for HCI in music.

Extensive analytical methods are used to look at corollaries of virtuosity and flow in real-world computer music interaction, using the example of *soundtracker* software as an alternative paradigm for computer-aided composition – a text-based notation that supports motor learning via the computer keyboard and a rapid edit-audition feedback cycle that helps to maintain focus and the *liveness* of interaction within the musical domain. To investigate the nature and development of skill and technique, look for evidence of flow experiences, and establish the use and

role of both visual and musical feedback in music software, the interaction and development of more than 1,000 sequencer and soundtracker users was recorded for a period of over 2 years. Quantitative analyses of interaction data are supplemented with a detailed video study of a professional tracker composer, and a user survey that draws on psychometric methods to evaluate flow experiences in the use of digital music notations, such as sequencers and trackers.

1.1 thesis summary

Creativity depends on experimentation. Experimentation requires support for sketching and fast working methods, facilitated by the skilled use of flexible tools. Tools that support sketching must support a high level of liveness, through fast feedback from the domain, but also balance the expressive power gained from abstracting aspects of the domain, in the interface and notation.

In music, there are many established abstractions, governing elements of music, as well as composition and production processes, which are prominent in the visually-mediated, metaphor-based interaction of popular modern music applications. The studio production process applied by sequencers and DAWs, for example, formalises a predominantly linear way of working that increases the viscosity of music editing; the act of recording commits the music to the notation, and acts as a watershed between creative and productive stages in the creative process.

By supporting virtuosity and flow, programs can extend creativity to computer-based interaction with a notation, such that flow is no longer limited to realtime performance with a musical instrument. By relaxing this requirement, interaction proceeds at a pace set by the user, making it easier for novices to maintain interaction flow, but also enabling experts to increase the complexity and scope of their musical expression. When manipulated through a device that supports motor learning, the program can support embodied interaction with the notation, and levels of immersion in music that are comparable to those found with live performance devices. Soundtracker software, through its use of a concise textual notation and computer keyboard, supports a rapid edit-audition cycle and demonstrates an example of virtuosity and flow in interaction with a digital music notation, lessons from which can inform the design of other music software.

1.2 aims and objectives

- To investigate user creativity in modern music software, and identify ways to improve support for a user's creative process.
- To evaluate the support of virtuosity and flow, as integral components of creativity, in interaction with digital notations.
- To work towards the development of a theoretical framework for the design and evaluation of notations in digital music.
- To study real-world interaction in soundtracking, as an example of virtuosity and flow in digital music interaction.
- To contribute to the limited canon of research on music composition processes and role of notation in creative music.

1.3 research contributions

The major contributions of this work to the field of music HCI are:

- *Theories and models.* Several models describing aspects of computer music interaction are proposed and evaluated, including:
 - a model of flow and liveness in musical systems (Chapter 4)
 - a descriptive model of music software interaction (Chapter 7)
 - a quantitative model of liveness in notation editing (Chapter 8)
 - a statistical model of flow in notation use (based on the *cognitive dimensions of notations* framework; Chapter 9)
- *Design guidelines.* A set of design heuristics for supporting virtuosity in the design of user interfaces, supported by empirical findings and other research.
- *Empirical methods and findings.* A new approach, combining several creativity research methodologies is developed, based on the longitudinal study of a large number of users, logging interaction in real-world creative scenarios, supplemented by psychometric-style surveys and a video study.

1.4 structure of dissertation

Following a description of relevant music technologies, in the next chapter; Chapter 3 reviews and integrates the limited catalogue of research in music composition, drawing on more general creativity research in psychology, to identify challenges faced in the design and evaluation of music software. As critical factors in creativity;

expertise and motivation are highlighted and discussed in the context of musical activities such as composition. Chapter 4 explores these factors in the context of the user experience, identifying design heuristics for supporting virtuosity and working towards a theoretical framework for modelling the creative user experience, based on *flow* (Csikszentmihalyi, 1996) and *liveness* (Tanimoto, 1990), subsequently used to support extensive user studies of interaction in computer-aided composition.

Chapter 5 outlines an experiment platform developed to collect and analyse data, capturing real-world examples of musical creativity from more than 1,000 tracker and sequencer users, over a 2-year period, employing a variety of empirical methods. Findings are presented in combination with a video study (Chapter 6), analysis of interaction logs relating to both motor skill in keyboard interaction (Chapter 7) and use of visual and musical feedback (Chapter 8), as well as user surveys that draw on psychometric approaches to develop a model of flow in notation use, based on the *cognitive dimensions of notations* framework (Green, 1996; see Chapter 9). Further to the general trends identified in this penultimate chapter, Chapter 10 concludes with a review of the findings, methods, and theories offered by this research, as well as the opportunities and implications for design and further study.



chapter two **sequencing, tracking, and the demoscene**

Buxton (1975) distinguishes between *computer-aided composition*, supporting a user's musical creativity, and *composing programs* that are themselves used to generate music. In contrast to trends in music research (Cascone, 2000),¹ this study focuses on the former, and technologies in mainstream music practices and aesthetics, such as amateur and professional music-making. The research also emphasises issues related to notation use in software environments, as supported by generic and already ubiquitous computer hardware (keyboard, mouse, screen), rather than specialist music systems.²

This chapter provides an overview of relevant technologies, beginning with the *digital audio workstation (DAW)*, the modern evolution of the *MIDI sequencer*,³ with which most readers should be familiar. Section 2.1 identifies the salient characteristics of their user interface and interaction, highlighting the role of performance capture, the use of *windows, icons, menus and pointer (WIMP)* in subsequent interaction, and the focus on production, in contrast to creativity (see Section 3.2) and composition (see Section 3.5).

Section 2.2 describes *soundtracking*, an alternative approach to computer music, based on interaction with a text-based notation manipulated using the QWERTY keyboard. This research uses the example of tracking to highlight factors in the computer music user experience that facilitate higher levels of focus and engagement,

¹ For example, where music programming languages (e.g. *CSound*, *Max/MSP*, *SuperCollider*) provide abstraction power that enables composers to innovate beyond what they see as limits in conventional musical formalisms, and below the level of the note (Desain *et al*, 1995; Cascone, 2000).

² MIDI devices are also discussed in the context of live performance. Multi-touch screens are becoming increasingly popular, allowing for wider adoption of new creative music environments, such as the *Reactable* (Jordà *et al*, 2005), and promising new ways to interact with notation-based systems. Though such emerging systems are not detailed, findings should be generalisable to these technologies.

³ *MIDI (musical instrument digital interface)* is a protocol for sending and storing music data (e.g. notes). A *sequencer* records and stores live data sent from MIDI-enabled instruments (e.g. keyboards).

and allow the development of virtuosity. Section 2.2.1 provides a technical overview of tracking, with details and examples of the notation, user interfaces, and specific packages. Section 2.2.2 also highlights the role of virtuosity beyond the user experience, within the tracker user community and wider *demoscene* sub-culture.

2.1 digital audio workstations (DAWs)

Modern *digital audio workstations (DAWs)* evolved from MIDI sequencer software, designed to capture and edit a performance from a MIDI instrument and later extended to include facilities to record and process audio (using acoustic instruments, mics, etc.). In this capacity, DAW software (such as *Steinberg Cubase*, *Apple Logic* and *ProTools*) not only integrates well with conventional, hardware-based recording studio practices, but is also used in smaller, more affordable home computer-based environments, as the foundation of the *desktop studio* (White, 2000).

high-level editing

The sequencer UI (Figure 1) revolves around the ① *arrange view* (sometimes called the *project window*), typically based on a linear timeline (Duignan, 2007), offering macroscopic views and editing of the song. In this window, blocks of music can be moved or copied, and subjected to high-level manipulations through automation envelopes that control global variables over time (e.g. volume, spatialisation, effect sends, etc.), but must be opened in a separate *device* or *part editor* to access and edit recorded content (e.g. notes, waveforms). Song data is entered into MIDI or audio tracks of data with the aid of the ② *transport bar*, which offers playback, spooling, and recording controls in the style of a tape recorder (Millward, 2005). Some further global and realtime track parameter editing and automation is possible through the ③ *mixer*, also styled after analog hardware.

low-level editing

To edit parts, a ④ *piano roll* (or *key editor*) plots pitch against time, often with an adjacent plot of volume against time, both of which enable manipulation of notes through the mouse.⁴ Sequencers also provide a ⑤ *score editor*, though Guérin (2004) highlights usability issues in creating a neutral, authoritative reference score from expressive and nuanced live interpretations.⁵

⁴ Interaction can be accelerated with the keyboard, but cursors are typically bound to existing notes or parts, requiring new notes to be drawn with the mouse or recorded through MIDI. It can also be difficult to select and edit objects that overlap (in time or pitch) using the mouse, or predict cursor behaviour, due to unclear orderings of events arising from nuanced timing in recorded performances.

⁵ In comparison to dedicated notation packages (like *Sibelius* or *Coda Finale*), the limited score editors in sequencers can be seen as best suited to quickly transcribing music for studio performers, supporting the recording process, rather than for the purposes of composition, playback, or typesetting (Wherry, 2009). Though this research does not look at score notation software in detail, many of the observations and findings in this report can be applied to these programs, which have also been observed to focus on *transcription* activities, rather than *exploratory design* (Blackwell and Green, 2000).

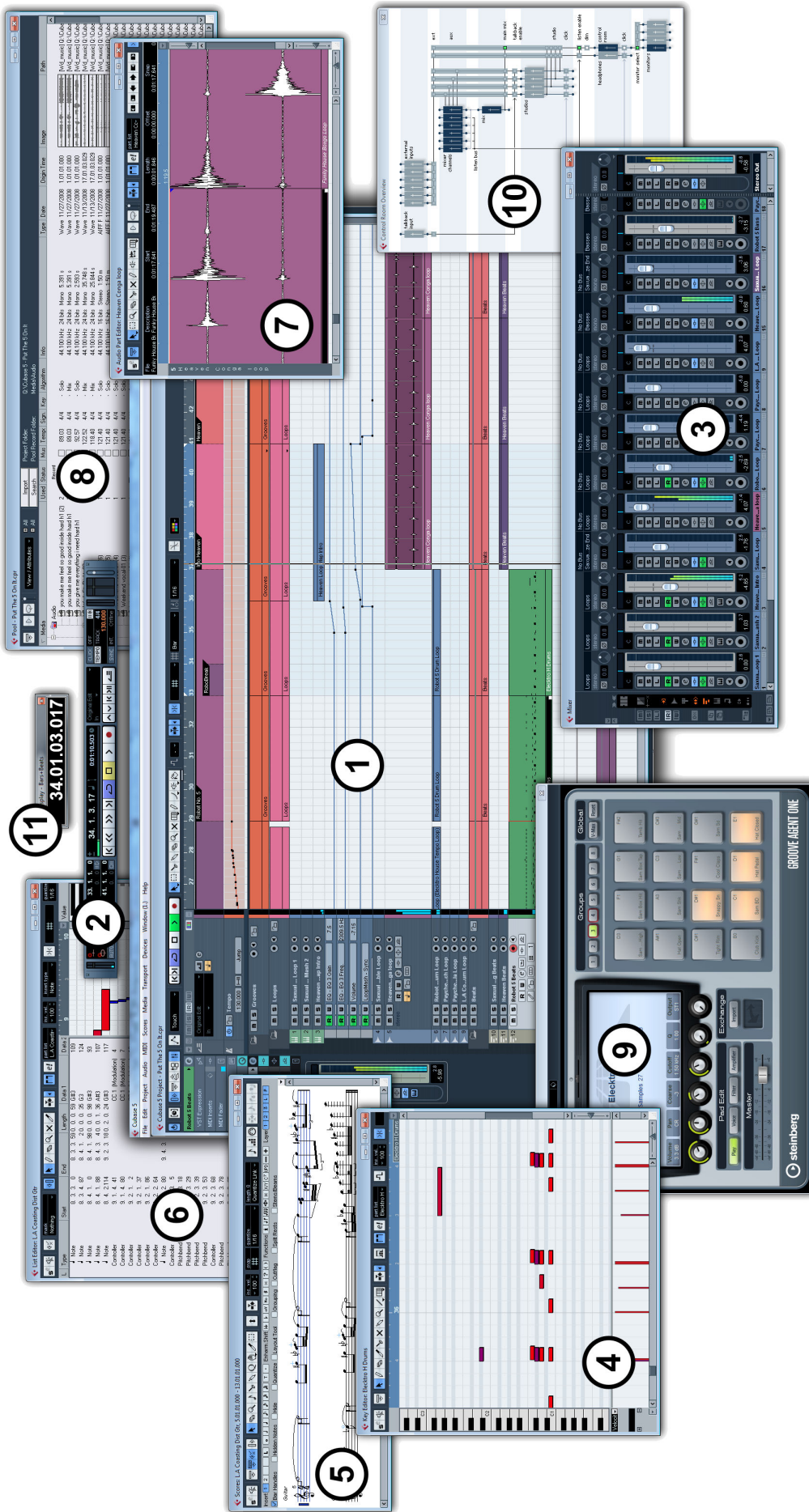


Figure 1 – Digital Audio Workstation (DAW). The common components within a modern sequencer / DAW (Steinberg Cubase 5 pictured):

- (1) Arrange / Project Window (including current track info; hierarchical track list; linear timeline with MIDI / waveform preview, in-place automation editing, and playback cursor)
- (2) Transport Bar
- (3) Mixer
- (4) Piano Roll
- (5) Score Editor
- (6) Data List
- (7) Audio Editor
- (8) Audio Pool
- (9) Plugin Synthesizer
- (10) Control Room Overview
- (11) Time Display / Timecode

sequencer
“devices”

digital audio

software synthesizer
and effect plugins

A number of parameters are not displayed or editable in the piano roll or score editor, but only found in the ⑥ *data list*, which provides an itemised list of all events in a MIDI part in text format. Millward (2000) observes that the relatively hidden nature of this data can lead to confusing program behaviour. Moreover, this highlights the use of multiple diverging layers of abstraction, in each device’s representation of the musical domain; different part editors offer different views and editing opportunities for different underlying data formats (waveforms, MIDI messages). Indeed, as either an extension or metaphor to the studio itself, it is thus possible to see sequencers as a container of connected but perceptually separate devices, rather than an integrated editing environment (Duignan *et al*, 2004).⁶

Programs thus also emulate the interconnects between devices, which route audio and MIDI signals around the system, but vary in their display of signal flow – from the “spaghetti hell” of dangling cables (see Figure 2(a)), to the absence of any visual cue in the mixer “sends” metaphor used in most programs (Duignan *et al*, 2004).⁷ The trade-off between the visibility and conciseness in showing such dependencies can present barriers to effectively using and understanding a system (Green and Petre, 1996).

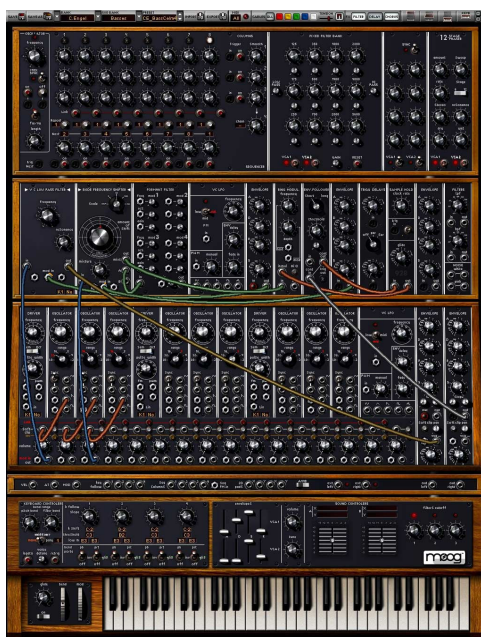
DAWs augment the basic sequencer concept with support for recording, importing, and editing of audio waveforms. Like MIDI, separate audio tracks share similar high-level preview and editing processes in the arrange window. However, the absence of note-level data limits editing audio parts. Instead, a basic ⑦ *audio editor* enables post-processing of recorded (or imported) waveform data. Unlike MIDI data, which is encoded in the project file, audio data is stored separately on the hard disk, and referenced as an external resource in the ⑧ *audio pool*.⁸

With the increasing processing power and decreasing latency of computers, DAWs extend their basic mixing and recording roles using *plugins* (e.g. ⑨), which provide software synthesis and DSP effect processing (White, 2001).

⁶ Chapter 4 discusses the implications of distributing interaction and domain representation across multiple redefinition sub-devices, rather than a comprehensive primary notation – insofar as they relate to facilitating flow, by dispersing focus and harming a user’s sense of control (see Section 3.7).

⁷ Figure 1 ⑩ shows a peripheral screen in *Steinberg Cubase* that illustrates a broad schematic of signal flow, between devices in the project. The UI of *Mackie Tracktion* (see Figure 5-16) explicitly enforces a left-to-right signal flow that also attempts to show effects, instruments, and routing in-place with the relevant MIDI or audio track.

⁸ While the portability of sequencer projects is also inhibited by the size and number of audio files, as well as the availability of hardware used, complex or unclear file dependencies significantly complicate management, copying, versioning, and backing-up (Duignan, 2007). This contrasts the integrated file formats used by trackers, which facilitate the sharing and revisioning of music (see Section 2.2.1).



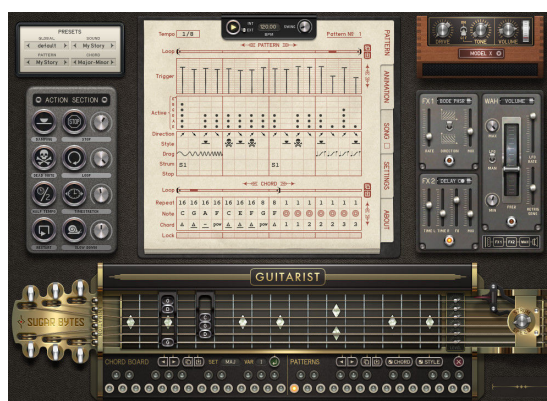
(a) **Virtual Analog Synthesizers**
(left) *Arturia Moog Modular V* plugin
(above) Original *Moog Modular* synthesizer



(b) **Rob Papen BLUE**
Promotional 3D perspective render of software-only synthesizer.



(c) **Luxonix Ravity S** plugin
Sliders, push buttons, and back-lit LCD displays, with GUI extensions to offer tabbing, menus, and patch lists.



(d) **Sugar Bytes Guitarist** plugin (above)
Virtual guitar and guitarist with rendered instrument and paper-effect step-sequencer, in 1940's aesthetic.

(e) **Antress Modern Series** plugins
Effects plugins styled as 1U and 2U rackmounted, outboard hardware, with analog pots, toggle switches, VU meters, LEDs, rocker switches, screws and vents.

Figure 2 – Sample images of DAW plugins.

Figure 2 shows a variety of plugin synthesizers and effects, again highlighting a trend towards visual metaphors to hardware devices. However, without the aid of a MIDI controller (which map generic physical buttons, pots, and sliders to controls in the software UI – see Figure 3), live and realtime control of these virtual devices can be cumbersome, especially using the mouse (Millward, 2004; Knörrig, 2006).

In practice, DAWs depend on specialist hardware; minimally a microphone or MIDI keyboard (White, 2000; Guérin, 2004; Millward, 2005), which enable the realtime entry of music or audio data. In contrast to the *windows, icon, mouse and pointer (WIMP)*-based editing of visual representations, these physical devices enable a more direct mode of interaction with music through acoustic or digital musical instruments, based on non-visual haptic or aural modes of feedback (Leman, 2008). The computer has only an incidental role during recording, as the user's focus rests entirely with the musical controller and performance. In this way, new musical ideas are established and explored through interaction with instruments (as in the past; see Graf, 1947; Harvey, 1999), and then subsequently committed to record, using a studio or sequencer (Boyd, 1992).

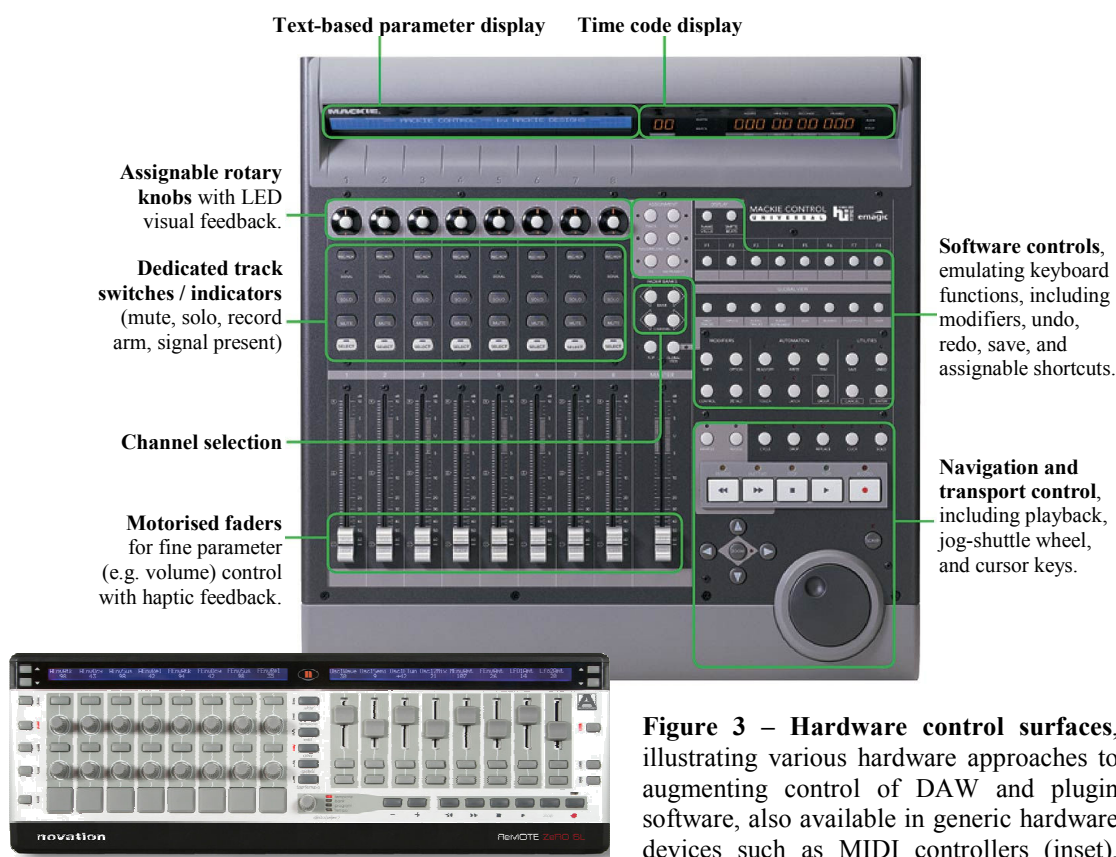


Figure 3 – Hardware control surfaces, illustrating various hardware approaches to augmenting control of DAW and plugin software, also available in generic hardware devices such as MIDI controllers (inset), showing assignable faders, pots, pads, etc.

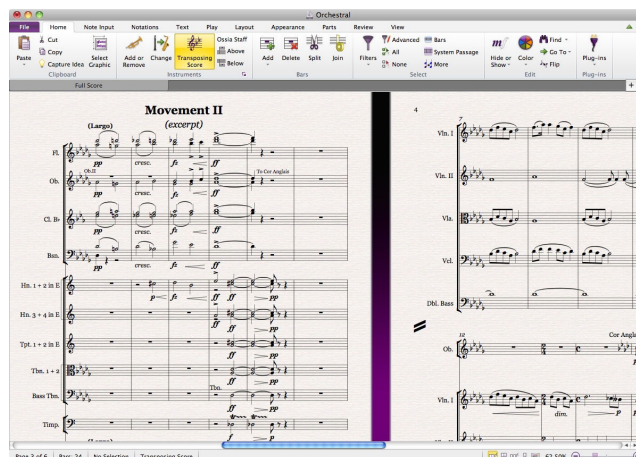
Recording acts as a watershed moment between creativity and productivity, such that the sequencer supports only the latter stages of a creative process; the final production and verification of an idea (Smith *et al*, 2007; also Duignan, 2007; see Section 3.2). Hardware interfaces that enable direct interaction with music (Figure 3) are designed for realtime use during recording, and are harder to exploit during subsequent “offline” editing.⁹

This research looks at the challenges of designing user interfaces in music software, examining the characteristics of musical performance and composition, to establish ways in which the computer can facilitate early-stage musical creativity without the recourse to analog methods, specialist hardware, or live performance seen in production software.

Other types of software also exist to support composition, but are not detailed in this research. Notably, *score editors* (Figure 4) are a logical evolution of the composer’s traditional use of score paper. However, they have only limited capacity to express digital music processes (Desain *et al*, 1995) and concentrate on visual, rather than aural, presentations of music. Blackwell and Green (2000) accordingly observed that their use seems limited to transcription (e.g. for performance), rather than as a medium for *exploratory design*.¹⁰ At the same time, the similarities between score editors, trackers, and sequencers mean many of the findings in this research can be generalised to other applications.¹¹

other software
for composition

Figure 4
Sibelius 7 (2011)
a comprehensive,
modern musical
notation editor



⁹ Exceptions to this include *control surfaces* (see Figure 3), *digital mixers*, and *hardware sequencers*, which have a wider remit in the production process, but which can also be seen to implicitly highlight limitations of software UIs by shifting key aspects of interaction to dedicated hardware.

¹⁰ This is reflected in the visual, document-based editing focus, *WIMP*-based interaction, and relatively limited sonification capabilities of notation packages, as suggested by the page layout view and *MS Office* style UI (including ribbon toolbar), in Figure 4. See also Sections 3.2 and 3.6 for the distinction between the creation of intermediary forms (sketches) and the production of a final manuscript.

¹¹ For example, like trackers and unlike DAWs, score editors present a single, central notation that likely benefits user focus (see section 8.4), and thus might support flow experiences if other criteria are also met (see section 3.7), through improvements in control (Chapter 7) and feedback (Chapter 8).

2.2 soundtrackers and the demoscene

This section begins with a technical description of tracker notation and interaction, followed by an account of the technological and social context, notably within the user community, and *demoscene* subculture, where the virtuosic coding and artistic talent of practitioners are as celebrated as the end product.

Figure 5
The tracker UI as demonstrated by *Impulse Tracker 2* (IT2), a 64-channel, 16-bit DOS tracker, developed by Jeffrey “Pulse” Lim from 1995 to 1999, styled on *Scream Tracker 3* (see Figure 8)

Including full-screen switchable tabs (with dedicated F-key) for:

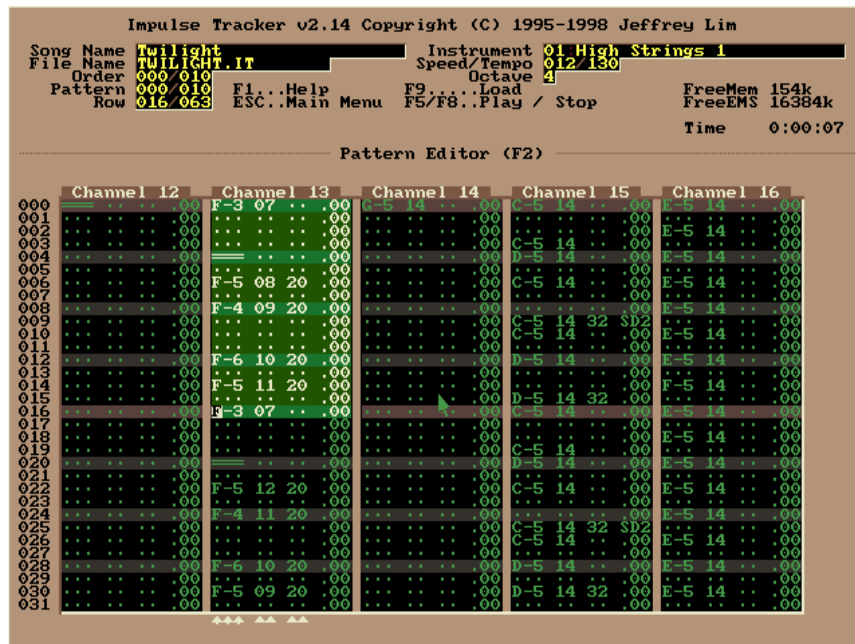
(a) Pattern Editor (F2) where music (notes & patterns) is edited;

(b) Sample List (F3) where samples are loaded and looped;

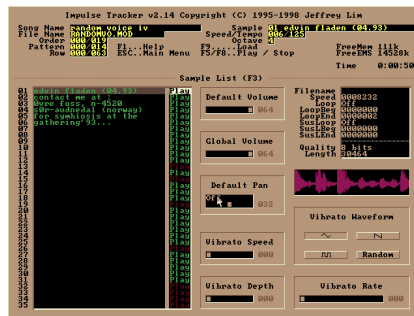
(c) Instrument List (F4) where samples are layered and filtered;

(d) Info Page (F5) offering an overview of song playback;

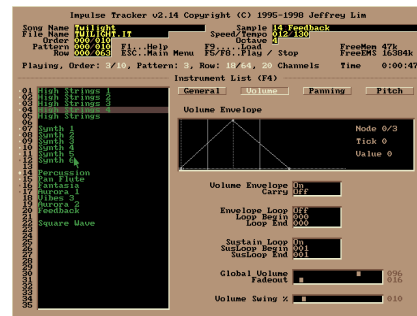
(e) Order List (F11) where patterns are ordered in the song



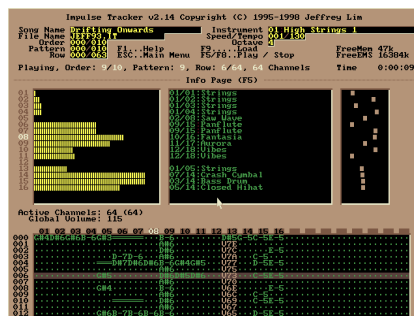
(a) Pattern Editor (F2)



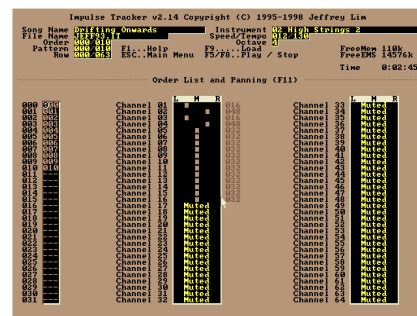
(b) Sample List (F3)



(c) Instrument List (F4)



(d) Info Page (F5)



(e) Order List (F11)

2.2.1 Technical Overview

Primarily using text to represent notational elements, a *soundtracker* (or simply “tracker”, pictured in Figure 5) allows the user to create *patterns* of note data comprising a short passage of music (often 4 bars). The music is realised in real-time, traditionally through an integrated sample engine and user-supplied set of samples. These patterns – resembling a spreadsheet in appearance, and analogous to a step-sequencer or player piano in function – are then arranged in a specific *order* to produce a *song*. The saved file (or *module*) stores the song together with all the notes, samples and instrument settings.

tracker notation

In the grid of the pattern, columns represent separate *tracks* (or *channels*) and the rows represent fixed time slices, like a step sequencer (see Figure 6). Each cell has fixed spaces for pitch, instrument, volume (or panning) and a variety of musical ornaments (or *effects*), for example: **C#5 01 64 D01** starts playing a note [C#] in octave [5]; instrument [01]; maximum volume [64]; with a slow [01] *diminuendo* [D]. Figure 6 shows an excerpt from a tracker pattern representing a single bar of music, inset with the equivalent phrase in conventional score notation.

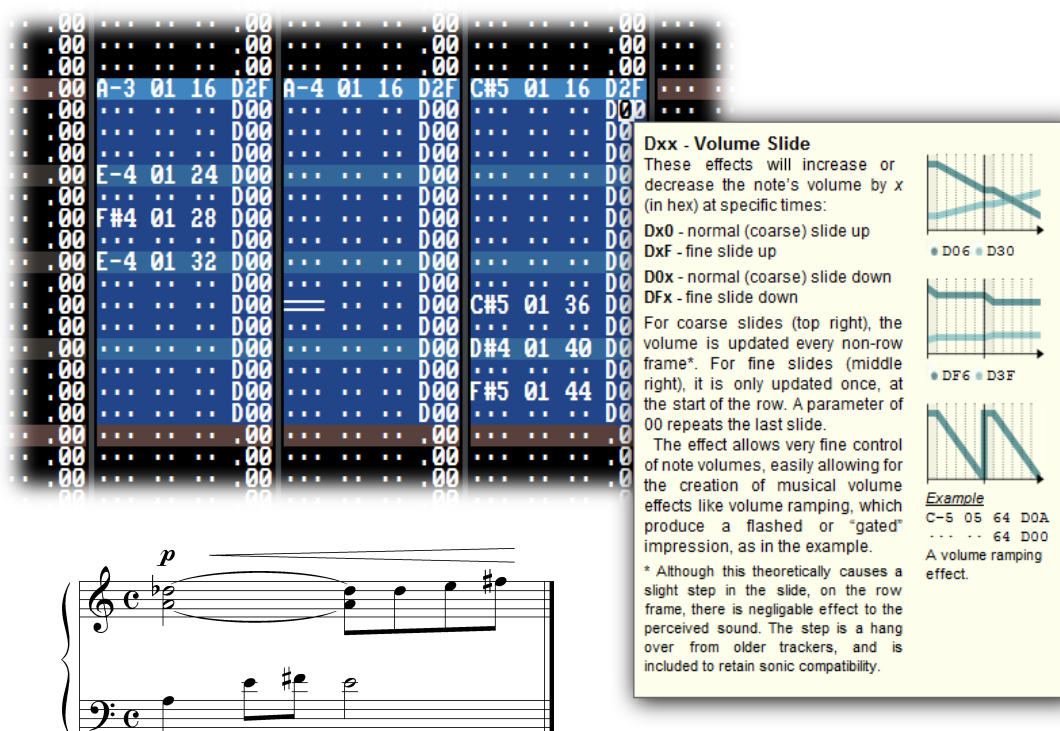


Figure 6 - Example of tracker notation, inset with equivalent score notation and overlaid with *reViSiT*'s in-program technical explanation of the effect used for the crescendo (DxF). Note how, in contrast to MIDI, which samples absolute values, tracker effects explicitly represent relative changes in musical parameters over time, similar to score notation.

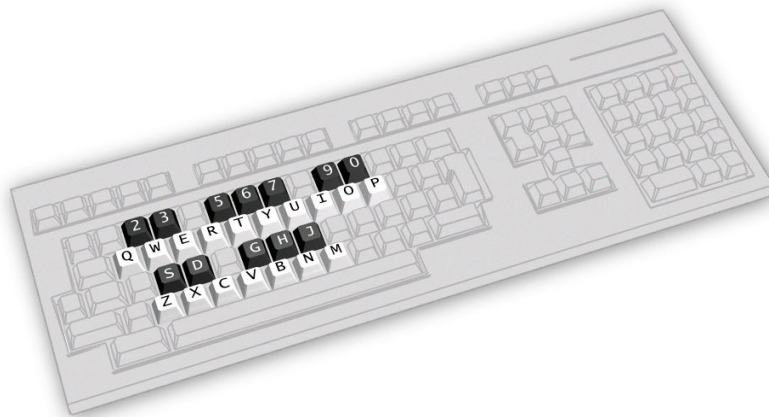
The final three digits in the cell enable a variety of other musical (and sound) effects, including other slides (e.g. portamento, glissando, filter, panning), oscillations (e.g. vibrato, tremolo), global variations in time or volume, or even branching in playback. These *effect* codes, while taking time to memorise, give experts a fast, flexible, and powerful way of adding musical expression to tracked music. Moreover, a number of effects also allow changes that are not easily expressible in score notation or MIDI, such as low-level control of sample playback or synthesis, which help bridge the apparent gap between a performer's control of sound and a composer's control of notes, and higher-level musical primitives found in notation (see Jordà, 2001). A full list of the effects available in modern trackers is given in Appendix A.

tracker interaction

Tracker programs are almost exclusively controlled using the computer's *QWERTY* keyboard, used for entry and editing of musical data, as well as management and navigation within the program. This allows the user to stay at the keyboard without incurring the time and focus penalties of homing between input devices, observed in sequencer use (Mohamed and Fels, 2002).

Figure 7

Note and pitch entry in tracker software. A two register, 29-key musical keyboard is superposed over QWERTY keys. Each register begins on C natural (Z / Q keys) and uses alternate rows of the keyboard for black and white keys (e.g. C# at S / 2 keys).



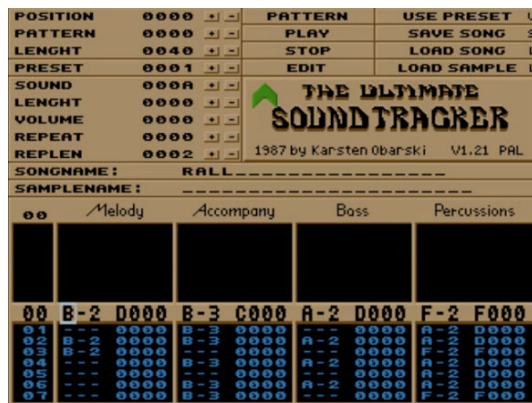
The text representation allows many parameters to be simply typed, but pitches are entered using a virtual piano (Figure 7). Interaction principally takes place in the *pattern editor* (Figures 5 and 8), and is mediated through keyboard shortcuts preserving visual focus on the notation itself. Shortcuts and macros accelerate all parts of the program, notably replacing the mouse's typical role in block selection and navigation, through the provision of rich-cursor movement. The cursor also plays a central role in triggering playback using the keyboard, allowing specific excerpts to be quickly targeted and auditioned.¹²

¹² The pervasive use and central role of the cursor in tracker interaction is explored further in Chapter 7.

Other parts of the program offer control of song, sample and instrument settings using more conventional interaction styles, such as buttons, sliders and text boxes, but typically play only a peripheral role after the initial set-up is complete. Nonetheless, these screens present fixed layouts and control focus, permitting the learning of screen configuration (as *spatial schemata*, see Section 3.6) to support fast visual inspection, navigation and editing using the keyboard.

graphical vs.
textual styles

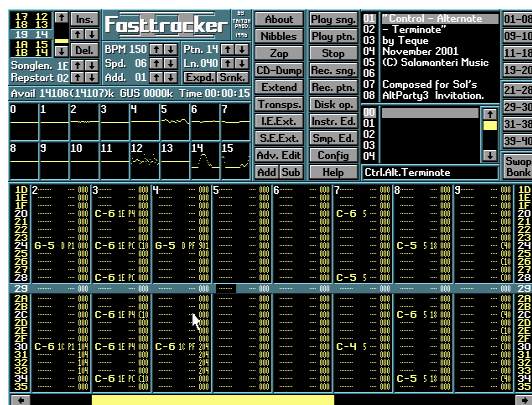
Figure 8 shows other tracker programs, from which two distinct design styles have emerged: a tiled, graphical UI designed for keyboard and mouse (*Ultimate Soundtracker*, *FT2*, *Renoise*); and a tabbed, text-oriented UI designed more exclusively for keyboard (*ST3*, *IT2* in Figure 5, and *reViSiT* in Figure 5-1).



(a) *Ultimate Soundtracker* (Amiga, 1987)



(b) *Scream Tracker 3* (DOS, 1994)



(c) *Fast Tracker 2* (DOS, 1994-1997)



(d) *Renoise* (Windows/OSX/Linux, 2002-)

Figure 8 – Notable tracker programs: (a) *Ultimate Soundtracker*, the original 4-channel, 8-bit tracker, by Karsten Obarski, released commercially by *Electronic Arts* for the Commodore Amiga in 1987; (b) *Scream Tracker 3* (*ST3*), the first major DOS tracker with 32-channel, 8-bit sample support, by Sami “Psi” Tammilehto, released by demo group *Future Crew* in 1994; (c) *Fast Tracker 2* (*FT2*), a popular 32-channel, 16-bit DOS tracker that competed with *IT2* (Figure 5), by Fredrik “Mr. H” Huss and Magnus “Vogue” Högdahl, released by demo group *Triton* in 1994; (d) *Renoise*, a commercial 32/64-bit tracker-based DAW, by Ed “Taktik” Müller and Zvonko “Phazze” Tesic (from code by Juan Antonio Arguelles “Arguru” Rius), released for Windows in 2002. See also Figures 5 and Figure 5-1.

comparisons
with sequencers

Summarising the tracker user experience, *Computer Music* magazine highlighted the virtuosity supported by use of the keyboard, observing, “The art of tracking has often been likened to a sort of musical touch-typing” (MacDonald, 2007). In a manner similar to expert programming editors such as *Emacs*, tracker programs avoid visual metaphor and graphical music notation abstractions, focusing on a concise textual representation and rapid manipulation of musical ‘source code’ – enabling quick edits, control of real-time interpretation (by synthesizer), and a *just-in-time debugging*-style mode of interaction (Nash and Blackwell, 2011; Church, Nash, and Blackwell, 2010; see Section 4.2.4).

As a tool for organising musical notes, trackers can be classified as a type of sequencer.¹³ Using the taxonomy developed by Duignan and Biddle (2005), Table 1 highlights similarities and differences in the user interfaces of related music software. While trackers use a textual medium, it lacks the descriptive power of freeform text, unlike live coding environments (e.g. *SuperCollider*). Moreover, the grid layout of the pattern shows the timing of events geometrically, similar to the graphical piano roll in sequencers. Patterns are thus subject to *eager linearisation* (events are shown in the order they are heard). However, the sequence of patterns can be affected at playback (with tracker effects, MIDI, or changes in the order list). This can introduce a delay to linearisation, with the advantage of increasing the *provisionality* of the music, as seen in loop-based programs like *Ableton Live* (Duignan, 2007).¹⁴ Like linear sequencers, however, the tracker editing focus is on notes, rather than loops or triggers, meaning they can be used for a broad range of musical styles. Ultimately, trackers can be placed on a continuum between linear and sample/loop-based sequencers, where distinctions and crossovers respectively help identify and generalise core properties of the computer music user experience.

	Linear Sequencers (Cubase, Logic, ProTools)	Soundtrackers (FT2, IT2, Renoise, reViSiT)	Sample/Loop Triggers (Ableton Live, FL Studio)
Medium	Graphical	Textual	Graphical
Abstraction	Predetermined	Predetermined	Predetermined
Linearisation	Eager	Eager/Delayed	Delayed
Event Ordering	Control	Control	Control
Applicability	General	General	Special

Table 1 – Characteristics of DAWs and trackers, based on Duignan and Biddle (2005)

¹³ *Renoise* describes itself as a “vertical” sequencer (see www.renoise.com). Similarities to step-sequencers, as well as early text-based sequencers, are also evident, especially in early trackers (see Figure 8). The German for tracker, *Rastersequenzer* (raster sequencer), also draws an analogy to raster graphics, based on the central role of grids (bitmap vs. tracker pattern) and sampling (sound vs. image).

¹⁴ Loop-based programs (like *Live*) focus on cycling audio loops, rather than longer passages of notes.

2.2.2 Social Context

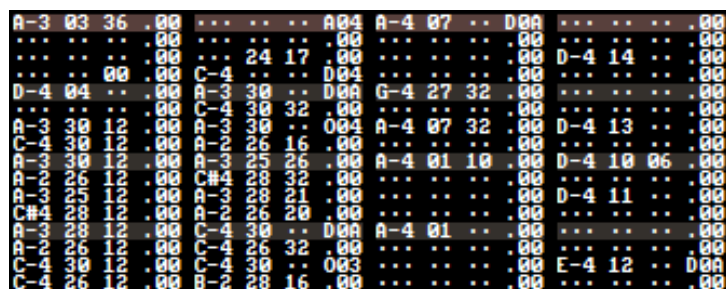
While this research focuses on the properties and design of the user experience, it is important to highlight the backdrop in which tracking developed. This section presents a historical overview of the users, developers and culture behind tracking, and the specific role of virtuosity, in the *demoscene* subculture.

*programming tools
for game music*

Emerging in the late 80's, the first trackers were based on technologies developed to provide music in computer games (Collins, 2008). They ran on home computers with very limited processing power, storage space, graphics, and audio hardware.¹⁵ Thus, the crude simplicity of the tracker's text-based interface and sound engine reflected not only its legacy as a programmer tool, but the limitations of the underlying hardware.

Consequently, the musical capabilities and appeal of early trackers appear limited: the early tracker *MOD* format supported 4 monophonic channels, hosting one of up to 15 instruments (mono, 8-bit, PCM samples). Sample compression, panning, and software mixing were beyond the hardware.¹⁶ However, for technically-minded users, such as young hackers and video gamers, these limitations simply presented a challenge, where they could compete against one another to defy the apparent limitations of the format. Tricks such as *polyphonic samples* (e.g. recorded major or minor chords) and *virtual polyphony* (interwoven monophonic pitches or samples to create the perceptual illusion of polyphony or multi-timbrality), enabled users to create rich musical soundscapes, and highly complex pieces in almost any musical style (e.g. Figure 9) – not just dance, but electronica, rock, jazz, blues, and even orchestral. The quality of the music improved, if not the sound quality, and user-defined samples offered significantly more sonic creativity and control than *General MIDI* sound sets of the time.

Figure 9 – Example bar from *Alternative Samba (1992)*, by Juha “Dizzy” Kujanpää. A 4-chn MOD, using all 31 samples (14 shown) with *polyphonic samples* and *virtual polyphony* to create the impression of two flamenco guitarists, keyboardist, and drummer in a jam session.



¹⁵ For example, the popular Commodore *Amiga* (the original platform of choice for the tracker musician) was a 16-bit computer with 20kHz stereo sound, typically booting programs from a 720KB floppy disk into 512kB of RAM.

¹⁶ Nonetheless, the programs represent one of the first examples of low-latency software synthesis, foreshadowing technologies now common in modern digital desktop studios (see Section 2.1).

Figure 10
Cracktro demo
from *Platoon*.



the demoscene

Ultimately a commercial failure owing to the perceived learning curve and chasm with conventional music paradigms, tracker programs became the care of these expert users, who continued development of the technology, as part of an artistic subculture called the *demoscene* (Tassajärvi, 2004; Polgár, 2008; Botz, 2011). This community of coders and artists began with young hackers, reverse engineering (‘cracking’) games to remove copy protection or change playing conditions. To flaunt their skills, hackers teamed together in *crews*, adding splash screens with credits, greetings, or messages to friends or rival crews (see Figure 10). Over time, these *intros* became a prominent showcase for coding talent, exhibiting increasingly complex visual art, animations, and music, ultimately eclipsing and displacing the original hacking activities.

demo parties

Practitioners and crews met at *demo parties*, partly to socialise, play games and swap coding tips, but mostly to exhibit or compete against each other with their latest demos and music. Works were judged not just on their artistic quality, but by the technical virtuosity shown by authors. Small gatherings that began in hackers’ basements have since grown into prestigious events, with tightly invigilated competitions, attended by thousands (Figure 11).

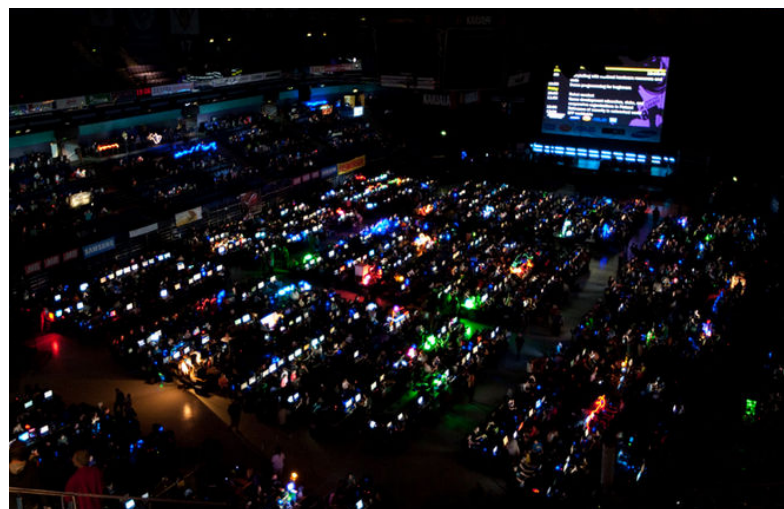


Figure 11
ASSEMBLY 2010
a modern demo party,
at the Hartwell Arena,
Helsinki, Finland.

Between parties and before the advent of the World Wide Web, crews would exchange, publish, share and review programs, demos and music over *bulletin board systems (BBS)*, or mailed floppy-disk compilations and newsletters (called *disk mags*) (Botz, 2011). The tiny file sizes of tracker music (typically around 100kB) enabled fast transfer over slow modem connections, yet contained all the music and sample data required for playback on any computer with a soundcard, without a need for specialist MIDI or audio hardware. Moreover, music was shared in a completely open format that allowed any listener to load, edit, change the music and re-use the samples in their own work. This opportunity represented a valuable learning resource for the novice, who could develop knowledge of the program by observing its use by others, tinkering with the music to learn the workings of the notation and program.

By the late 90's, the scene had moved to the IBM PC and DOS, and new programs and formats began to support up to 64 polyphonic channels of CD quality audio. However, at the same time, the original, increasingly obsolete MOD format survived. Although modern DSP, resampling, and upsampling marginally improved the playback quality, the format was retained specifically for its limitations and its capacity to test the ingenuity of composers, whose endeavours to defy listeners' expectations and garner the respect of their peers continued.

Similar trends are witnessed across the *demoscene* (Botz, 2011). *DirectX* and video hardware provided similar leaps in graphics capabilities for demos, but competition categories formed with explicit limits on the size of demos, allowing entries no more than a 64kB (or even 4kB) footprint¹⁷ for their executable – which must contain the entire code and content for the presentation of all visual and audio content (text, textures, graphics, animation and camera scripts, music, sound samples and synthesiser). Exploiting complex *procedural generation* algorithms to mathematically create intricate textures, shapes, visual effects, sounds and music, winning entries nonetheless deliver intricate, high quality, high resolution audio-visual spectacles, often several minutes in duration, and comparable to other productions that are measured in megabytes or gigabytes. Figure 12 shows stills from a recent 64kB demo placed 2nd at *Assembly 2011*, with a description of the techniques used.

In the 20 years of tracking history, though hardware capabilities and user interface design have moved on, the basic design of tracker software and interfaces has changed little - in appearance,

¹⁷ For comparison: today's average web page is 320kB [Source: Google].

function, or use. Little effort has been made to make the programs easier-to-use, and obsolete file formats and limitations from DOS and Amiga programs are still supported and celebrated. Instead, the appeal of these programs and their limitations are that they specifically provide a challenge to the user; that mastering them is rewarded both intrinsically and extrinsically, in the user experience and community respectively. Accordingly, the designers of new musical interfaces should consider more than the simple sonic capabilities or usability factors of their innovations.

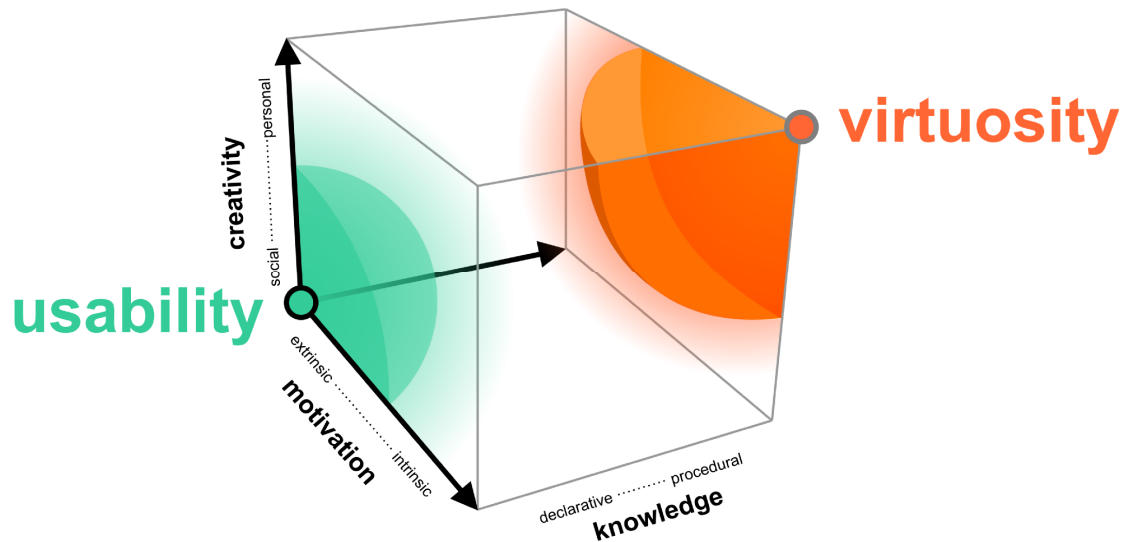


Figure 12 – *Uncovering Static* (2011), by Fairlight + Alcatraz.¹⁸ 59kB demo with over 5 minutes animation (shown in 1080p) and symphonic techno soundtrack.

Graphics: realtime rendering (no pre-calculation), distance field manipulation, procedural generation (spores, buildings, textures, clouds), ray-casted ambient occlusion (lights, shadows), Boolean algebra, and post-processing (filters).

Audio: MIDI song data with realtime physically-modelled instruments (solo and layered strings, piano, oboes, breath pad, cymbal, orchestral and rock bass drums and snares) and analogue synthesis (lead and pad), plus effects (reverb, water-like LPF).

¹⁸ Executable available at www.pouet.net/prod.php?which=57449 (HD video also available on YouTube)



chapter three **creativity, expertise and motivation**

This chapter presents an overview of creativity research and theory from the fields of psychology, HCI, and music, using it to explore the challenges of user experience design in music, specifically music composition based on notation use.

Over the course of the chapter, an argument for supporting virtuosity and flow is outlined, entailing three broad shifts away from the principles of usability design, as illustrated above, outlining an effort to move beyond the productivity of current software methods, towards interfaces that support creativity.

Section 3.1 presents a definition of *personal creativity*, such as that in expressive arts, crafts and music, rather than the social creativity in problem-solving environments, such as business, science or professional design. Section 3.2 considers models of the creative process that can inform the design of the user experience, highlighting the limits of working with formal notations. Section 3.3 establishes expertise and motivation as the key factors of creativity; acknowledged in musical practices, but often deprecated in usability techniques. Then, treating the user interface as a creative environment, Section 3.4 explores strategies to facilitate creativity, taken from psychology research.

(continued overleaf)

The final three sections respectively explore the issues of creativity, expertise and motivation in the contexts of music and the computer. Section 3.5 looks at creative processes in music composition, and the roles of sketching and performance. Section 3.6 looks at the nature of expertise, the role of procedural knowledge (such as motor skill) and declarative knowledge, and their respective emphases in music and HCI. Lastly, Section 3.7 looks at the sources of motivation in creativity, music, and the UI, highlighting the importance of an intrinsic reward in creative activities, and looking at Csikszentmihalyi's theory of "flow" as a framework for combining concepts of creativity, expertise, and intrinsic motivation, which is used later to develop a model of the creative user experience in music (see Chapter 4).

3.1 personal creativity

In researching creative practices, it is crucial to define the context that creativity is studied in (Hewett *et al*, 2005). Depending on the individual, domain, or research goals, definitions of creativity vary significantly, with implications for findings and practice (Blythe *et al*, 2007). At the same time, apparent differences can arise as a result of terminology, stemming from a “parochial” tendency to study creativity within individual domains (Wehner *et al*, 1991). This section draws on several dimensions identified in psychology research (Mayer, 1999), developing a definition of creativity that can be applied to the design and evaluation of the user experience, specifically as concerns creative authoring tools and software in music. Table 2, at the end of the section, summarises the definition of creativity adopted here, as a specific intersection of this “n-dimensional taxonomy” (Hewitt *et al*, 2005).

*creativity =
novelty + value*

Mayer (1999; Table 1) demonstrates a broad consensus among researchers that creativity is the production of something *novel* (*new, original*), but which also has *value* (*appropriate, significant*) within a given context. This definition fits with the wider public’s implicit concepts of creativity (Sternberg, 1985), but becomes increasingly complicated when the various contexts of creativity are considered (Mayer, 1999; Hewett *et al*, 2005).

Table 1 – two defining features of creativity
(adapted and extended from Mayer, 1999)

author	feature 1	feature 2
Gruber and Wallace (1999)	novelty	value
Martindale (1999)	original	appropriate
Boden (1999, 2004)	novel	valuable
Nickerson (1999)	novelty	utility
Sternberg and Lubart (1999)	novel	appropriate
Amabile (1983, 1996, 2006)	novel	appropriate
Mayer (1999)	originality	usefulness

*P-creative
vs. H-creative*

Boden (2004), who explores computational approaches to navigating and mapping creative domains, at the same time makes an important distinction between novelty as recognised by the practitioner and that of history itself: *P-creative* describes acts that have a personal, subjective significance and should be understood in a *psychological* context; whilst *H-creative* describes objective novelty, the impact of which can be understood in *historical* and *social* contexts. Other researchers have made similar distinctions (Maslow, 1968; Sternberg, 2003; Hewett *et al*, 2005). Csikszentmihalyi (1996), for example, uses a ‘big C’ to denote socially-validated creativity, discussing subjective, inter-subjective and objective evaluations. Others further distinguish between novelty within a specific social-group or society and humanity at large (e.g. Mandler, 1995; Nickerson, 1999).

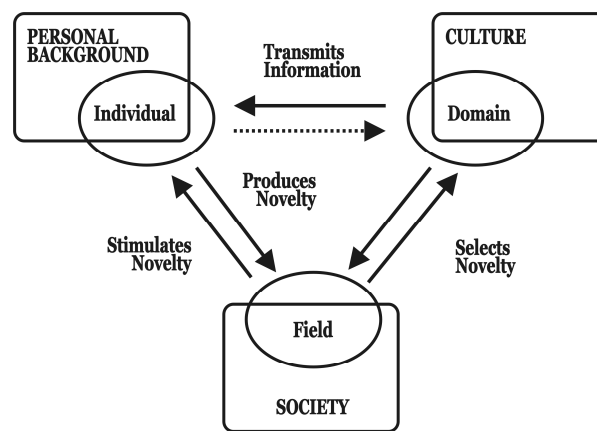
P-creative
 \subseteq *H-creative*

*the role of
 recognition*

Naturally, these classifications are not mutually exclusive. H-creative, for example, is a subset of P-creative, since something novel to everyone is invariably novel to its creator (Boden, 2004; Hewett *et al*, 2005), even if they don't themselves fully-recognise the value of the innovation (Shneiderman, 2002).

Social recognition also comes after the creative act, upon dissemination of a work (e.g. through a concert, exhibition, or publication). Some researchers class the act and art of persuasion as an integral component of being creative (Sternberg, 2003; with Lubart, 1995). Others see it as a final stage that can be considered separately, after-the-fact (Simonton, 1994; Shneiderman, 2002).

Figure 1 – a systems view of creativity
 (Csikszentmihalyi, 1999)¹



*the individual,
 field, and domain*

In his *systems perspective*, Csikszentmihalyi (1999; Figure 1) contends that Creativity is “the ability to add something new to the culture”: where culture comprises a variety of *domains* (cultural or symbolic spaces), which an *individual* transforms or extends, but which is guarded by a *field*, whose members (e.g. teachers, buyers, and critics) act as gatekeepers that evaluate and admit new ideas.²

*primary
 vs. secondary*

In an opposing perspective, Maslow (1963) maintained that studies should not focus on the outcome of creativity, but on the process. For the individual, novelty and value is apparent during the act, notably as a motivating factor (Sternberg, 1985; with Lubart, 1995). Like Boden, Maslow (1968) describes two kinds of creativity: *primary*, pursued for self-actualisation and fulfilment; and *secondary*, leading to the sort of achievements recognised by a field. Sternberg (2003) notes that the latter is the kind “with which scholars in the field [of creativity] are more familiar”.³

¹ Figure 1 is as cited, with minor addenda: the arc **Individual** → **Domain** is de-emphasised, to reflect the gate-keeping role of the **Field** (Csikszentmihalyi, 1999), but retained to show the broadly-perceived attribution of creativity; the triad is re-oriented to facilitate comparisons with later work (Chapter 4).

² Csikszentmihalyi (1996), however, notes that social validation may not come during the individual's lifetime, citing Van Gogh as someone who only posthumously became “Creative”.

³ A focus on recognised works is also tacit in specialist domains like music (Collins, 2005), which tend towards studies of outcome and achievement rather than process, as discussed in Section 3.5.

*the individual's
perspective*

Maslow's approach, however, appears to resonate more with creative individuals themselves (Boyd, 1992; see section 3.5), where artists see their activities as acts of self-expression, rather than efforts to please a crowd. Discussing strategies to support creativity, Nickerson (1999) argues,

[H]owever we conceive of creativity we should not make its existence dependent on it being recognized as such. By definition, we are not aware of creativity that goes unnoticed, but we have every reason to believe that it exists.

HCI researchers must thus be careful of how they conceive of creativity, and how they use a social or systems perspective to that end, as it requires one to "recognize the fact that the audience is as important to creativity as the individual to whom it is credited" (Csikszentmihalyi, 1999). It is doubtful such a view is held by many artists (Sloboda, 2005; see section 3.5); and thus a systems perspective would seem to conflict with any mental model we might hope to develop for artistic users. Even the most groundbreaking Creativity seems not to be motivated by social efficacy, but by an intrinsic interest in the activity itself (Csikszentmihalyi, 1996). Section 3.7 specifically details how social factors can have an adverse effect on an individual's motivation and creativity by involving ego, harming confidence (e.g. to take risks), and encouraging conformity (Amabile, 1983; Nickerson, 1999), to the extent that artists tend to explicitly seek isolation and solitude (Getzels and Csikszentmihalyi, 1976; Boyd, 1992).

*active vs. passive
social involvement*

At the same time, whilst an individual might not actively engage with the field, no one can absent themselves entirely from the system (Fischer, 2005). The domain in which an individual works is defined by its field, who not only determine the works in wide circulation, but also the commonly-used notations, symbols, tools, and practices (Csikszentmihalyi, 1996; Fischer, 2005). Even a self-taught musician can't escape the influence of what they hear on the radio, nor the fact that the design of their instrument has likely evolved from centuries of historical precedent. In this sense, "Whether one follows the crowd or takes a different path, it is usually impossible to ignore what takes place in the field." (Csikszentmihalyi, 1996) A critical distinction, however, is how free an individual is to follow different paths, in relation to how constrained they are to follow the field.

*the field
of interaction*

In digital creativity, the influence of the field is prominent in the UI, which shapes a user's view of a creative domain, as might otherwise be the role of mentors, tutors or other peers (Lubart, 2005), defining the methods, representations and interpretations

possible. A degree of formalism is inherent in the digital extension of any domain (Dix, 2005), but compounded by the need to generalise and standardise a user experience for multiple users, often drawing on, and thus perpetuating, existing conventions and formalisms in a field (Kitzmann, 2003; Blackwell *et al*, 2008).

Music, especially, is a field where well-established formalisms and traditions (see section 3.5) not only serve to operationalise the musical domain (Johnson-Laird, 1988), but can also constrain an artist's creativity (Sloboda, 1985; Boyd, 1992). Yet, while score notation grew out of a need to formalise music so it could be communicated to other musicians, the page also supported informal, personal notes and the sketching of ideas in forms that need only be understood by the composer (Graf, 1947; Schubert and Sallis, 2004). Section 3.5 explores this affordance and the role of sketching in creativity, highlighting the limitations of music software based in formal music systems.

creative genius

Another consequence of taking a sociocultural perspective of creativity is the resulting focus on eminent creative practitioners that make the most impression on society, and thus a tendency to describe creativity as the exclusive province of genius (Weisberg, 1993; Csikszentmihalyi, 1999).⁴ However, if there's one benefit to the increased availability, affordability, and accessibility of powerful creative tools, such as the computer, it's that a wider proportion of the population has the means to engage in the creative activities (Resnick *et al*, 2005; von Hippel, 2005).

the average case

Often seen as the father of scientific creativity research (Plucker and Renzulli, 1999), Guilford (1950) described creativity as a "pattern of traits that are characteristic of creative persons", and stated that "[w]hatever the nature of creative talent may be, those persons who are recognised as creative merely have more of what all of us have." His *psychometric* tests were designed to efficiently and economically measure creativity quantitatively, supporting comparisons between genius and the 'average case' (Sternberg 2003; with Lubart, 1999).⁵ However, the limited insight offered by such paper-and-pencil approaches (see Amabile, 1983; Gruber and Wallace, 1999) prompted a shift "from the measurement and development of presumably general underlying traits of creative ability toward analysis and explanation of remarkable instances of real-world creative accomplishment." (Feldman, 1999)

⁴ For example, Leonardo, Mozart, Darwin, Michelangelo, Einstein (in Simonton, 1994, 1999; Csikszentmihalyi, 1996; Mayer, 1999; Shneiderman, 2002; Boden, 2004).

⁵ Psychometric research, which monopolised early studies of creativity, developed in parallel to studies of intelligence (Plucker and Renzulli, 1999), itself originally motivated by the desire to find, measure, explain, and cultivate "genius" (Albert and Runco, 1999).

*prodigious and
innate talent*

Weisberg (1993) cautioned against limited “genius only” views, which lead to the assumption that successful creativity depends on innate talent or personality traits that individuals are born with, as evidenced by child prodigies.⁶ Howe (1999) concludes,

As far as the likelihood of someone eventually becoming capable of mature creative accomplishments is concerned, the fact that one was a prodigy in childhood is significant not because it points to some inherent special quality of the person, but simply because it provides an indication of significant progress having been made while the person was still young.

*developing
creative ability*

Ericsson *et al* (1993) similarly demonstrated that what people have previously considered “innate talent” is more accurately explained as the result of many years of disciplined and deliberate practice. Correspondingly, the development of expertise is now widely seen as central to creativity (Amabile, 1983; Sternberg, 2003; see Section 3.3). Mastery of a tool (e.g. a musical instrument) allows the individual to more effectively explore and focus on the domain (see Section 3.6). However, the threshold at which expertise enables creativity is much lower than that entailed by social recognition, which would otherwise prevent us from calling most children creative, and so present problems for studies of creativity in education or other development environments (Barrett, 2005; see also Section 3.5).

*everyone can
be creative*

Boden (2004) observes that creativity draws largely on “everyday psychological abilities, such as noticing, remembering, and recognizing” that can be specialised for skilled application within a given domain, with effort. Sternberg (2003; with Lubart, 1995) similarly sees creativity as a “decision” – a willingness to be creative, and a choice to invest the required effort to develop relevant domain skills, cultivate patience and open-mindedness, and learn to question the status quo.

Such modern theories of creativity (see also Amabile, 1996) champion the notion that, with effort and inclination, anyone is capable of great creativity, and all that is required is the right environment to bring this forth (Feldman, 1999; Howe, 1999). These environments are defined by one’s surroundings, teachers, families and communities (Barrett, 2005), but increasingly also one’s computer, which like the others must encourage (or avoid discouraging) an individual towards development and personal growth, and balance formal learning with independent thinking and experimentation, in order to facilitate creativity.

⁶ For example, in music: Mozart, Chopin, Mendelssohn, and Paganini (Graf, 1947; Harvey, 1999).

Recently, the advancement of communication and data storage technology (notably the Internet) has led to the wider availability and retention of ‘lesser’ creative works, presenting further opportunities to study amateur, non-professional, and less-recognised creativity (Bardzell, 2007). As in studies of creativity in education, research into less exceptional achievements and talents provides insights into creative individuals in the formative stages of development (Barrett, 2005), as well as the creative opportunities that would appear to lie within reach of a far greater cross-section of society (von Hippel, 2005). In this capacity, the *demoscene* and *tracking* sub-cultures in computer music that arose in the 1980s and 1990s, supported by early connectivity technologies (such as *bulletin board systems*, see Section 2.2) act as an early example of the democratisation of creativity enabled by the computer. Similarly, this research also represents a study of real-world creative activity, in tracking and sequencing, and development that is not limited to recognised practitioners, but also includes complete beginners (see, for example, Chapter 5).

Shneiderman *et al* (2005) also embrace the Internet as an opportunity to move beyond what they see as creativity research’s focus on the individual as solo practitioner, advocating software support for creative collaborations. While acknowledging a social context, even Csikszentmihalyi’s systems perspective revolves around the individual (Barrett, 2005; Fischer *et al*, 2005).⁷ Support for creative thinking in project teams working on specific problems appears an achievable and valuable goal (Shneiderman, 2002), but there seems an inherent conflict between the introverted nature of artists and the extroverted nature required in working with others (detailed further in Section 3.7).

The solo life of artists needn’t be a consequence of limitations in working methods or technology, but possibly a conscious choice and desire of the artist. Beyond *homage* or loosely-linked ‘schools’, social collaboration would appear to be the antithesis of what artists are seeking when they look to isolate themselves (Getzels and Csikszentmihalyi, 1976).⁸ For individuals with mature skills, collaboration can be highly productive and enjoyable, but entails an objective (or inter-subjective) social perspective of creativity. As with recognition (see earlier), the presence of other people in the creative process must also be

⁷ Fischer (2005) also suggests how to extend a systems perspective to group and distributed activities.

⁸ In popular music, collaborative songwriting is much rarer than the sheer number of groups suggests; bands can be dominated by one member, take it in turns to write songs, or have a separate songwriter, and often break up as a result of differing personal sentiments, high emotions, “bad chemistry”, or simply the desire to pursue solo projects (Boyd, 1992).

considered for its capacity to negatively impact an individual's motivation, freedom, and confidence (Runco and Sakamoto, 1999; see 3.7), especially when they are still in the process of developing skills (Collins and Amabile, 1999; Nickerson, 1999; see 3.6).

Moreover, while the goal of distributing the creative process across multiple participants will hopefully become increasingly relevant, this project aims to demonstrate that HCI, at least with regard to music, faces more immediate challenges in supporting an individual user's creativity that should be addressed before considering more complex scenarios.⁹ Fischer *et al* (2005) state,

[D]espite the inherent social aspect of creativity, individual knowledge, imagination, inspiration and innovation are the bases for social creativity; without inspirational sparks from the individual, social creativity simply has no chance to flare up in the first place. Augmenting and then better utilizing individual creativity is thus essential for achieving social creativity.

One of the first objectives of HCI research into creativity should therefore be to “enhance the personal experience” (Hewitt *et al*, 2005). Accordingly, in the context of this research, the social dimensions of activities such as musical performance and improvisation are not explored in detail.¹⁰ Terms such as ‘performance’ and ‘audition’ are used in a technological context only, more generally referring, respectively, to any time-critical execution of a task or direct interaction with a domain (cf. “Level 4 liveness”, discussed in Section 4.2.4) and realtime evaluation of a creative product, irrespective of social context. In this sense, a performance does not imply the presence of listeners other than the practitioner (e.g. audience or collaborators).

Table 2 summaries the implications for defining creativity in this context, and as it relates to HCI, as discussed in this chapter. The next section proceeds with a review of research on the mental processes involved in an individual's creative process, in an attempt to carry forward such knowledge into the design of the user experience.

⁹ Indeed, the desktop studio of sequencers and DAWs (see Section 2.1) can already be seen as a model of a collaborative environment (i.e. the electronic recording studio), which might offer an explanation for interaction issues that arise in solo use (such as focus, discussed in Chapter 8). In this sense, whilst art and creativity research have focused on the individual, music software has had an implicit tendency to focus on collaboration, even as they look to target individual users.

¹⁰ For deeper discussions of social aspects of music interaction and perspectives on how technology can facilitate group-based music interaction, group flow, collaborative musical creativity and performance liveness, see Sawyer (1995, 2006), Bryan-Kinns (Nabavian and Bryan-Kinns, 2006; Sheridan and Bryan-Kinns, 2008; Bryan-Kinns and Hamilton, 2009), Auslander (1999), and Emmerson (2007).

property of	products – persons – processes
Creativity is studied by looking at the processes that underpin the creative user experience, where a user (creative individual) uses a notation to create new data (creative product). (see Section 3.2; Ward et al, 1999).	
significance	personal – social
The user experience is judged on its capacity to support self-expression, self-actualisation and fulfilment, as judged by the user themselves. (see Section 3.7; Maslow, 1968; Nicholson, 1999)	
creative ability	universal – special
Users are not assumed to have innate creative talent or genius, but the ability to develop creative ability and motivation. Everyone can, to some extent, be creative; which computers can facilitate (or obstruct). (see Section 3.6, Nicholson, 1999; Weisberg, 1999).	
enabling skills	domain-specific – general
Whilst the user may have general creative skills, the UI governs how a (symbolically-encoded) domain is manifest in the user experience, and how skill is developed within it and the domain (e.g. music). (see Section 3.6; Csikszentmihalyi, 1999; Ward, Smith and Finke, 1999; Boden, 2004)	
measurement	qualitative – quantitative
Both qualitative and quantitative methods are used to study aspects of creativity, such as motivation and expertise, to provide a balanced account of both the process and product of creativity. (see Chapter 5-9; Hewett et al. 2005; Collins, 2005, 2007)	
activity	individual – collaborative
The research focus is on single-user systems, and explicitly explores the advantages arising from the perceived lack of collaborators or observers during the creative act. (see Section 3.7; Amabile, 1983; Hallam, 2002)	

Table 2 – the definition of creativity applied in this research (using the n-dimensional taxonomy proposed by Hewett *et al*, 2005; based on Mayer, 1999). (with references to discussion in the text and supporting literature)

3.2 the creative process

stage theories
of creativity

In crafting a user experience to support creativity, UI designers have a significant influence over the creative process – not simply by defining the explicit interactive procedures of the creative act, but also shaping the environment’s inherent capacity to support creative thinking and working styles (Nickerson, 1999; Auh, 2000; Hewett *et al*, 2005). This section explores models of the creative process, whilst illustrating how only the final, incidental stages of creativity are supported by many modern user interfaces, even in the creative arts, and general HCI approaches to usability.

It is evident in even the earliest theories of creative thinking that creativity draws heavily on unconscious cognitive processes (Martindale, 1999). In 1896, Hermann von Helmholtz reflected on his own creative process, identifying three stages: *saturation*, *incubation* and *illumination* – in which, respectively: an individual familiarises themselves with the details of a challenge; waits for the mind to reconcile them; until the moment a solution becomes apparent. In 1908, Henri Poincaré described this process as a period of conscious thinking about a problem, followed by a period of unconscious thought, until a solution bursts back into the conscious mind, after which deliberate work is undertaken to verify the insight. Their reflections were formalised by Wallas (1926) as one of the first models of the creative process, based on discrete stages of unconscious or conscious thought: *preparation*, *incubation*, *intimation*¹¹, *illumination*, and *verification*.

problem solving
vs. problem finding

Nickerson (1999), however, associates stage-based, step-wise models and their descriptions exclusively with problem solving, as faced by scientists, mathematicians, or philosophers, which can be difficult to apply to less well-defined creative activities, such as artistic self-expression. Gruber and Wallace (1999) also note a tendency to focus research on problem solving activities, encouraged by the relative ease of studying and modelling well-formed, easily-articulated tasks.

Artistic expression is as much about finding problems as solving them (Getzels, 1975; Alty, 1995; Collins and Amabile, 1999; Runco and Sakamoto, 1999; Sternberg, 2003). An artist does not set about a given, well-defined problem, nor pursue a definitive solution that pertains to truth or correctness; their intentions are ill-defined and changeable (Collins, 2007; see Section 3.5), and the merit of their solutions is subjectively good-or-bad, and only right-or-wrong in a socio-cultural context, itself changeable.

¹¹ The *intimation* stage, characterised by the feeling of an impending breakthrough, is often omitted in more recent accounts (e.g. Webster, 1989) and seen as a component of other stages.

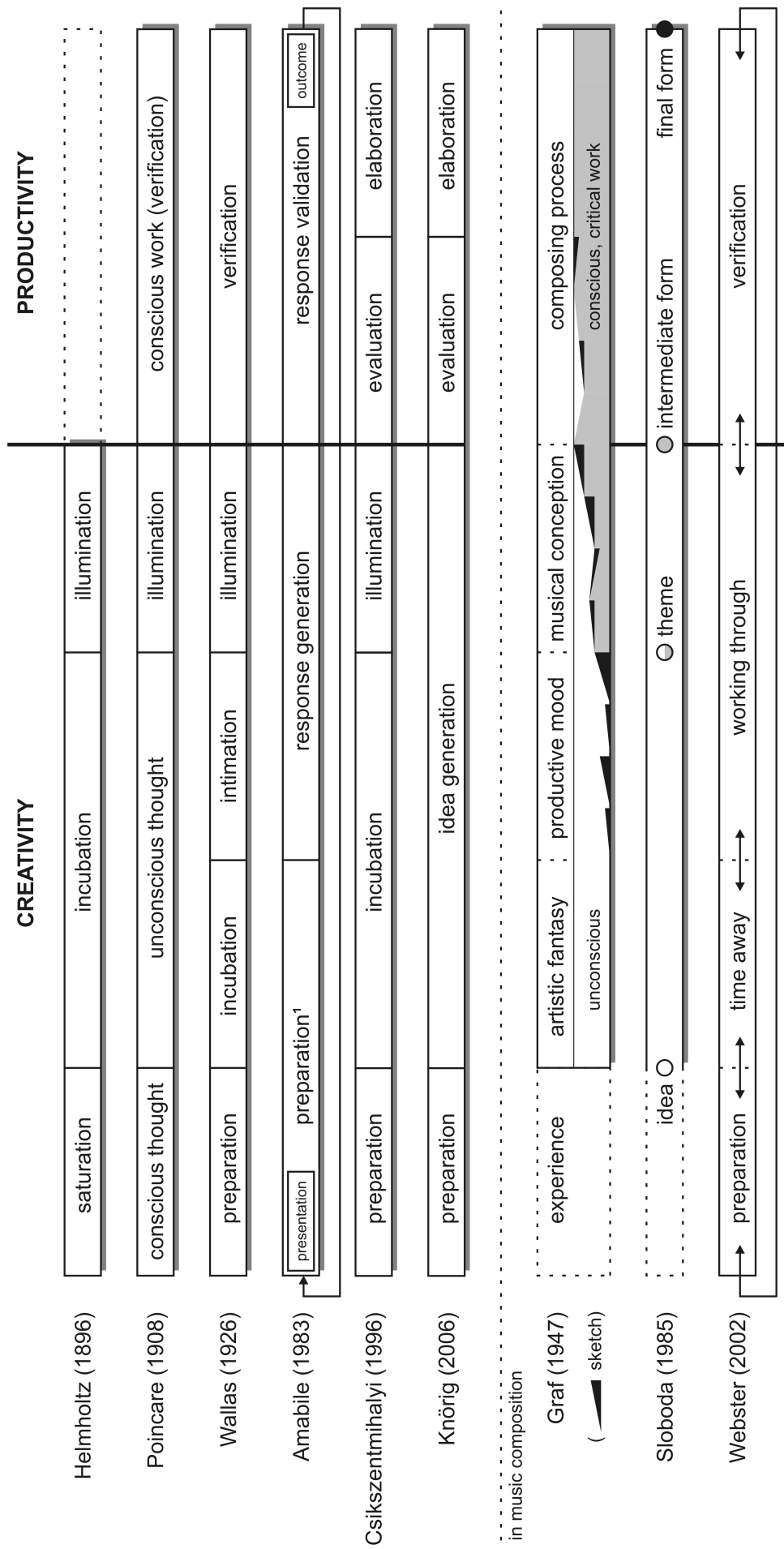


Figure 2 - a comparison of stage-based theories of the creative process, and three descriptions of the music composition – within the “creativity” and “productivity” phases of “innovation”, as characterised by Amabile (1996). The diagram also includes an illustrative example of the role of sketches, in Graf’s composition process.

¹ Amabile (1983) defines part of this stage as “reactivating” information, implying the re-emergence of non-conscious knowledge.

Composer Jonathan Harvey also appears to have trouble relating his ‘inspiration’ to Wallas’ model, observing that there are opportunities for inspiration at almost every stage of the composition process, including the revisions prompted by verification (Delige and Harvey, 2006). Accordingly, music researchers have attempted to adapt stage theory for application to the expressive arts: looking for more complex, non-linear routes through the stages (Burnard and Younger, 2002), more iterative and recursive applications of the process (Webster, 1989, 2002; Knörig, 2006), or parallelised, potentially ill-defined, problem-solving activities at different granularities of detail (Collins, 2005). As such, stage theory remains relevant in modern research (Martindale, 1999; Burnard, 2007; e.g. Csikszentmihalyi, 1996; Shneiderman et al., 2005; Knörig, 2006); especially in music (see Sloboda, 1985; Webster, 1989; Auh, 2000; Burnard & Younger, 2002; Delige and Harvey, 2006; Collins, 2005, 2007).

Figure 2 illustrates several of the popular models of stage-based theories of creativity process, including three representations of the musical composition process (as described by Graf, 1947; Sloboda, 1985; Webster, 2002) that illustrate the difficulties music researchers can have trying to reconcile less-structured musical creativity with stage-based models.¹²

Highlighting the limits of stage-based theories, Weisberg (1993) and Burnard (2007) both argue that, while unconscious processes play an important role in creativity and are difficult to articulate, there is no reason to believe we are not in control of them, and that they are not driven, overlapped and more finely interwoven with conscious thought.

In *psychoanalytic* theory, Kris (1952) described *primary* and *secondary* cognitive processes respectively corresponding to an unconscious, dream-like mental state characterised by concrete images, and a conscious, waking, abstract state characterised by logical reasoning.¹³ Creative thinking, he argued, involves *adaptive regression* – the ability to alternate between the primary state, where new thoughts are formed, and the secondary state, where they are elaborated. Accordingly, Kubie (1958) talks about an intermediate *preconscious* state, which contains unconscious thoughts that can be subjected to conscious interpretation.

¹² Few accounts of the composition process explicitly define a conscious, preparatory stage of musical creativity, highlighting the lack of a well-defined problem that the composer seeks to solve. Section 3.5 explores this stage of the creative process in the context of developing music skill and knowledge, which allows musicians to navigate the musical domain without defining an end-goal (see Alty, 1995).

¹³ Not to be confused with Maslow’s *primary* and *secondary* creativity (see Section 3.1).

emerging awareness

Reviewing the habits and accounts of notable composers, Graf (1947) offered a description of the creative process in musical composition as it was before the age of computers. He describes three broad phases in musical creativity, as listed in Figure 3.

Figure 3 - three phases of creativity
(Graf, 1947)

1. *preliminary work done by the unconscious*¹⁴
2. *combined work of unconscious and conscious mental powers*
3. *conscious final polishing of the form*

In this process, the secondary processes of the conscious mind and critical thinking gradually take a larger role, as artistic fancies and fantasies are formed by the primary processes of the unconscious. While Graf's description of the composition process is less tied to discrete stages and offers only a loose description and chronology of different moods and moments in the process, the broad stages of Wallas' model emerge from his account (Deliege and Harvey, 2006).

using sketching to probe the unconscious

Crucially, Graf (1947) also identifies the *musical sketch* as a practical mechanism used by composers to mediate between conscious and unconscious thinking – illustrated in Figure 2 as separate, smaller, and shorter creative processes that allow composers to experiment with ideas that may (or may not) contribute towards the final form of the music, culminating in a draft that is finalised using critical thought alone. More detailed discussion of musical creativity, drawing on Graf's accounts of composition and the role of sketching, is given in Section 3.5.

goal-oriented vs. exploratory creativity

Ward, Smith and Finke (1999) characterise the difference between problem solving and problem finding approaches as “goal-oriented versus exploratory creativity”. This begins to mirror terminology found in HCI literature, where Hewett (2002) notes a similar tendency towards formulating and solving well-defined, testable goals, operators and methods as efficiently as possible, rather than the open-ended exploration critical to creativity. Candy and Edmonds (2004) similarly argue that problem finding and problem solving require different skills and thinking styles, but only the latter seems supported by technology.

divergent and convergent thinking

Instead of a procession of distinct stages, the creative process can be characterised as a broad arc over *divergent* and *convergent* thinking styles, where several ideas are generated, then selectively pursued or elaborated, based on perceived merit (Sternberg and

¹⁴ Graf uses the term “subconscious” for this phase, also drawing on Freudian psychodynamic theory of the early 20th Century, in his account of artists' lives. As Graf correctly predicted; “It is probable that new research will alter the Freudian constructions” (Graf, 1947, p80). Even so, though Freud himself had deprecated the term (preferring “unconscious”) and though theory has since moved on, the change in thinking does not otherwise invalidate Graf's broader description of the composition process.

Lubart, 1999; Plucker and Renzulli, 1999). The significance of divergent thinking was the major finding from psychometric studies of creativity (e.g. Guilford, 1950), in which *ideation* (the quantity, rather than quality, of ideas) was measured and linked to higher creative performance (Gruber and Wallace, 1999).

“wicked” problems

As an exercise in problem-solving, McBride and Brown (2007) suggest that creative self-expression might be regarded as what Rittel and Webber (1973), in the field of social policy, refer to as a “wicked” problem – an ill-defined, ongoing challenge with no definitive or standard solution (see Figure 4). By contrast, they consider well-defined tasks and challenges, such as those focused on by creativity and HCI researchers, as “tame” problems.

Figure 4 - the ten characteristics of a “wicked” problem
(Rittel & Webber, 1973)

1. *There is no definitive formulation.*
2. *They have no stopping rule.*
3. *Solutions are not true-or-false, but good-or-bad.*
4. *There is no immediate and no ultimate test of a solution.*
5. *Every solution is a "one-shot operation"; there is no opportunity to learn by trial-and-error, every attempt counts significantly.*
6. *There is no enumerable (or an exhaustively describable) set of potential solutions, nor is there a well-described set of permissible operations that may be incorporated into the plan.*
7. *Every problem is essentially unique.*
8. *Every problem can be considered to be a symptom of another.*
9. *The existence of a discrepancy representing a problem can be explained in numerous ways. The choice of explanation determines the nature of the problem's resolution.*
10. *The planner has no right to be wrong.*

“wicked” art

Applied to music, these characteristics mirror recent adaptations of stage theory that introduce iterative or cyclic processes to tackle ill-defined problems.¹⁵ They also implicitly identify the benefits and roles for technology in facilitating creativity. For example, whilst live performance (and recording) might be considered a “one-shot operation”, technology offers us the chance to immediately review, and optionally revisit, material.

¹⁵ Most creative domains have established sets of permissible operations; accepted techniques or systems, such as schools of painting, rules of harmony, tonality, etc. Formal notations can thus be seen as part-formulations of a problem, allowing solutions to be codified and enumerated. Lerdahl and Jackendorff (1983) have even theorised about identifying a formal grammar for Western tonal music, albeit with limited practical success (Cross and Woodruff, 2009). Indeed, celebrated creativity is often characterised by bending or breaking rules (Simonton, 1994). For example, at the beginning of the 20th Century, some composers pronounced the exhaustion of the major/minor key system, in Western tonal music, inspiring the members of the Second Viennese School, such as Schoenberg, Berg and Webern, to develop new tonalities, e.g. atonalism, serialism, chromaticism (Grout and Palisca, 1996).

Beyond the social context, this gives an individual the opportunity to learn by trial-and-error, and the “right to be wrong”.¹⁶ This role of immediate feedback, in facilitating iterative sketching of musical solutions highlights how the computer can reduce the wickedness of a creative problem without subtracting from the scope or reward of the challenge, is a major focus of subsequent theoretical (Chapter 4) and empirical work (Chapter 5 to 9).

beyond elaboration

Born from a mathematical heritage, HCI and computing can be seen to tend towards precise definitions and the solving of well-formed problems (Hewett, 2002; Candy and Edmonds, 2004; McBride and Brown, 2007). Problem solving methodologies assume that the user has “collected” all the ideas already, and merely has to converge on the final solution (Shneiderman, 2002). In design scenarios, and especially in music, the individual is thus expected to already know their production goals and strategy before sitting down in front of the computer. For example, section 4.2.3 demonstrates how the dominant model for music software, the sequencer, relies on a recording paradigm that largely depends on the user having already prepared a musical performance beforehand, not necessarily through the computer.

Existing authoring software is thus often exclusively able to support the final stages of creativity: the elaboration, verification or refinement of an idea, “and take the ‘aha’ moment of insight as having already happened” (Smith *et al*, 2009). In music, sequencers and DAWs vary dramatically in their support for experimenting and playing with new musical ideas. Music programs that seek to support *exploratory design* are often used simply for *transcription* (Blackwell and Green, 2000), where the user experience becomes one of productivity rather than creativity (Knörrig, 2006), both of which are integral parts of the innovation sought by individuals and organisations (Amabile, 2006).

¹⁶ Roberts (2000) outlines three general strategies to tackle “wicked” problems: *authoritative*, where the number of stake-holders is reduced; *competitive*, where parties are pitched against each other; and *collaborative*, where parties work together to find a mutually-agreeable solution. As such, the focus on personal creativity (see Section 3.1) might be seen as an “authoritative” strategy to tackle creativity; in comparison to the competitive, social, or collaborative approaches also discussed.

3.3 creativity synthesized

As suggested in the last section, art presents “wicked” problems for practitioners, researchers, and toolmakers (Rittel and Webber, 1973). In natural and social sciences, individuals pursue these problems for palpable rewards, such as new technologies, lower crime, or better education. In professional and social creativity, they pursue the expressed needs and desires of a client, society, or culture, for rewards of money, recognition, or fame. However, in personal and artistic creativity, practitioners pursue their own objectives and set their own challenges, often without the promise of rewards from others. The emphasis shifts from the expertise needed to solve problems, to the motivation needed to find them.

creativity
= *expertise*
+ *motivation*

This section explores the interdependence of expertise and motivation, as integral components of creativity (Amabile, 1983, 2006; Sternberg, 1999, 2003; Boden, 2004). Expertise and skill, especially in music, are detailed further in Section 3.6. Likewise, Section 3.7 explores motivation, specifically *intrinsic motivation* (where the activity is its own reward – Amabile, 1983), and the theory of “flow” (Csikszentmihalyi, 1990, 1996), which unifies intrinsic motivation and skill development.

Domain knowledge and expertise is widely seen as critical to creativity (Ward, Smith and Finke, 1999; e.g. Amabile, 1983; Nickerson, 1999; Boden, 2004).¹⁷ Originally seen as an innate talent, the ability to be creative is now more widely linked with acquired skill, gained through work or practice over an extended period (Ericsson *et al*, 1993; Weisberg, 1999). In either socially-recognised or culturally-relevant creativity, knowledge of a domain is required to identify the opportunities for novelty and the techniques to produce it, both of which require great expertise, developed over many years of effortful learning (see Section 3.6). Boden (2004) observes that “the more impressive the creativity, the more expert knowledge is typically involved.”

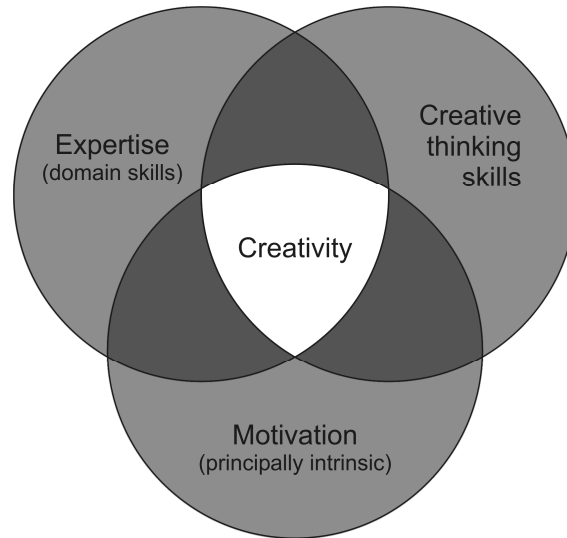
personal
development

In personal creativity, there is no predefined level of challenge or skill – they are determined by the individual themselves. Whereas an expert brings their considerable skill to bear on difficult, socially-significant activities, a novice derives a similar sense of personal achievement from tackling tasks that are easier, but which make no less a demand of their relative ability (Csikszentmihalyi, 2000). An individual’s satisfaction with their achievement motivates them to seek new tasks, in which they look for a greater challenge, comparable to their growing experience

¹⁷ See also Ward, Smith and Finke (1999, p207-8) for further references.

and ability (Hallam, 2002). In this way, expertise is developed gradually, and motivation is maintained *intrinsically*, with less dependence on uncertain and often limited external validation and support (Collins and Amabile, 1993; see Section 3.7).

Figure 5
Componential
Model of Creativity
(Amabile, 1996, 2006)



*Amabile's
componential
model of creativity*

In both personal and social environments, creativity can thus be seen to depend on both expertise, to meet the challenge of being creative, and motivation, to pursue both the challenge and the acquisition of expertise (Csikszentmihalyi, 1996; Nickerson, 1999; Sternberg, 2003; Boden, 2004). Amabile (1983; see also 1986, 1996, 2006) explicitly combines these factors in her *componential model of creativity* (Figure 5), in which she describes creative performance as the confluence of: domain-relevant knowledge and abilities (or expertise, detailed in Section 3.6), intrinsic motivation (detailed in Section 3.7), and creative thinking skills (described below).

Figure 5 illustrates creativity as the intersection of three inter-related dependencies – that is, all three are requisite for creativity itself, but also depend on each other. For example, one must be motivated not just to be creative, but to develop expertise (e.g. to practise), where such learning can itself be a strong motivator, by stimulating and empowering the individual (Candy and Edmonds, 2004). Similarly, creative achievements that are enabled by expertise can provide further motivation to develop new skills and pursue further creativity (Hallam, 2002).

*creative
thinking skills*

The last component, “creative thinking skills”, denotes cognitive styles and attitudes that determine how flexibly and imaginatively people approach problems (Amabile, 2006). As presented in other confluence theories of creativity (Sternberg, 2003; with Lubart, 1995), they involve willingly embracing concentration, focus,

effort, energy, complexity, ambiguity and rule-breaking, as well as knowledge of work styles, techniques or heuristics that are likely to provoke original thinking, but which might otherwise be counterintuitive. These traits are not only a product of an individual's personality, but also affordances of the creative tools and environment, which may support or hamper such skills.

Unfortunately, many of these attitudes are actively discouraged in computing and mainstream HCI theory and practice (Gentner and Nielsen, 1996). Many usability textbooks (e.g. Nielsen, 1993; Shneiderman and Plaisant, 2005) explicitly oppose complexity, ambiguity and effort. Attention and focus is divided in multi-tasking, multi-window environments; concentration is broken by errors, notifications, or background processes; complex processes are automated, hidden, abstracted, and simplified, obviating challenge and effort; rule-breaking is defeated by pre-defined and well-defined interaction procedures (e.g. wizards), notions of "correct actions"¹⁸, and preset tasks or templates; and ambiguity or uncertainty is frustrated by the precision demanded explicitly by formal notations and implicitly by requirements of digital encoding. Related issues are highlighted in the context of music software, such as DAWs and sequencers (described in Section 2.1), in the remaining sections of this chapter and in subsequent chapters on empirical findings (notably, Sections 8.4 and 9.3).

3.4 supporting creativity

The more we understand the creative process and mind, the more we can identify factors in the experience or environment of the individual that contribute towards creativity, over which we may have some influence (Nickerson, 1999).

Components such as expertise and motivation, highlighted in the last section, were recognised as ingredients to creativity in the 1950's, when psychometric tests were developed to identify the creative potential of children, in an effort to ensure 'gifted' individuals received appropriate support and encouragement (Plucker and Renzulli, 1999). Whether one accepts creative skill as a product of nature or nurture and whether there exists a limit to one's creativity or not, Nickerson (1999) argues that almost everyone has the potential to be more creative, and that "few people realise anything close to their potential in this regard."

¹⁸ As explicit in HCI evaluation methodologies that require precise and correct interaction steps to be defined, such as *Cognitive Walkthrough* (Polson *et al*, 1992) or *GOMS* (Card *et al*, 1983).

measuring
the benefit

In the 1960's, when modern creativity research was still young, several speculative theories for enhancing one's creativity (notably Edward de Bono's "lateral thinking") achieved popular success, with the public and organisations, but showed only mixed results in subsequent empirical tests (Nickerson, 1999).¹⁹ This, however, raises the practical difficulties in defining and measuring such an abstract and disputed quantity as creativity (see Csikszentmihalyi, 1999, Gruber and Wallace, 1999), which make it difficult to rigorously evaluate the effectiveness of any given strategy (Nickerson, 1999). For example, controlled experiments cannot provide a free and open environment for creativity, but represent contrived or constrained scenarios that can be hard to translate to the real-world (Hewett *et al*, 2005; Collins, 2005).

objectivity
vs. subjectivity

Maslow's humanistic approach explores creativity from the perspective of the individual, rather than from that of their peers or society (see Section 3.1). In modern creativity research, this approach is largely deprecated, due to the perceived lack of scientific rigor in the qualitative assessment of subjective phenomena. With reference to Maslow's view, Csikszentmihalyi (1999) accepts the importance of an individual's subjective experience, but argues that a systems perspective is necessary for the objective, scientific study of creativity and its cultural impact (see also Barrett, 2005; Hewett *et al*, 2005).²⁰

idiographic
vs. nomothetic

In creativity research, Simonton (1999) observes that many studies of the creative process, which include biographical approaches, observation, or detailed case studies of unique individuals and acts, are necessarily *idiographic* (based on single or small samples). While such studies can be revealing (see Section 3.5), it can be difficult to ascertain the broader relevance of findings concerning inherently unique individuals and processes (Gruber and Wallace, 1999). At the same time, the labour involved in such longitudinal studies can make it difficult to extend them to larger samples of individuals.

Science, he argues, is better served by *nomothetic* approaches (generalised explanations, based on large samples), such as his own historiometric analyses of creative output (Simonton, 1980, 1984, 1994; also Section 3.5). However, such studies reveal little

¹⁹ Sternberg and Lubart (1999) are similarly sceptical of such unproven approaches, and voice concern about their ensuing rise as the public face of "creativity research".

²⁰ It can also be seen to retrospectively vindicate the tendency, in research, to focus on the lives, works, traits, and environs of a minority of highly successful, influential, and recognised creative individuals (see Webster, 1989; Wehner *et al*, 1991; Albert and Runco, 1999; Bardzell, 2007). Though his earlier works (such as flow theory, see Section 3.7) focus on studies of individuals from various backgrounds, Csikszentmihalyi (1999) cites his more recent attitude as a reaction to the frustrations he had in attempts to rigorously study largely subjective experiences and processes.

about the creative or cognitive processes involved in the production of a work. Moreover, only the extant creative output of historically-recognised individuals can be studied (Hall and Sallis, 2004). Simonton (1999) argues that this implicit vetting by society and history ensures a work is “unquestionably” creative.²¹ At the same time, whilst a larger sample size is used, it is inherently biased towards a minority of creative practice, and history offers little opportunity to verify trends found in famous individuals, against a control group representing the wider population. Sternberg and Lubart (1999) instead call for new scientific approaches that mediate between the narrow focus of psychometric and experimental approaches and the narrow validity of biographic approaches.

*the creative process
as user experience*

In contrast to the field of creativity research, HCI has drifted beyond its own ‘system’-focused perspective towards individual-oriented, *user-centred* methodologies for the evaluation and design of user interfaces (Sharp *et al*, 2007; e.g. Norman and Draper, 1986). These approaches accept the user experience as a subjective one, but one that can be usefully, if not always rigorously, tested with qualitative methods such as field studies that might include observation, user surveys, and interviews. Shneiderman *et al* (2005) also note the value of multi-dimensional, in-depth, longitudinal case studies.

*studying creativity
via the computer*

Researchers have advocated the use of the computer in studies of creativity: to facilitate larger and more complex studies of creative output (Simonton, 1994); to augment subjective feedback from the individual with objective measures, and reduce the interruption or distraction inherent in self-reporting (Collins, 2007; e.g. Sloboda, 1985); to remove the researcher and observers as intruders in the creative process (Perkins, 1981; Collins, 2005); or as a platform for setting, controlling or measuring the creative task or environment.²² Candy and Edmonds (2004) recommend practice-based research, which combines technology-based art projects with observational activities (e.g. Nash and Blackwell, 2008), mirroring the standard approach of researchers in the field of music instrument design, as in the *NIME (New Interfaces for Musical Expression)* conference series (a musical offshoot,

²¹ As such, Simonton (1994) is in accord with Csikszentmihalyi’s position that the scientific study of creativity inherently depends on social validation. However, he explicitly sees the act of persuasion as separate from the creativity itself. Csikszentmihalyi (1999), though, dismisses this strategy as “epistemologically” problematic, arguing that, under his definition of creativity, even an individual cannot be sure of their own creativity, if they can’t persuade someone else of the fact. This reasoning, however, would seem to presuppose that artists, and humans at large, always think rationally.

²² See Scripp *et al* (1988), Webster (1989), Kratus (1989), Smith and Smith (1994) or Folkestad (1996).

originally a workshop, of leading HCI conference, *SIGCHI*), which focuses on novel technologies for interacting with music.

Within this artist-led field, practitioners often feel confined by notations and visual UIs in software (especially in usability-oriented, end-user GUIs), favouring more direct interaction with sound, offered by live performance with specialist hardware enabling more embodied approaches (e.g. gesture, haptics) (Leman, 2008). The use of digital archiving (recording) processes can also be seen to further obviate the need for notation, while shifting the focus of musical creativity from composition to improvisation (further discussed in Section 3.5).²³ Moreover, the methodological challenges of studying creativity have limited use and development of techniques and theory for evaluating new interfaces with respect to usability, ergonomics, or expressivity. It is hoped that this study of notation-based computer interaction and related theoretical considerations (Chapter 4) can facilitate wider discussion of these topics in the field of music research.

In developing creative support tools, Hewett *et al* (2005) advocate a mix of subjective, qualitative, or idiographic methods and objective, quantitative, or nomothetic methods – specifically citing ethnography, computer logging, or participatory design, as methods that more richly capture the creative user experience, in comparison to traditional HCI approaches that use performance metrics and largely concentrate on user productivity.

Accordingly, the research in this thesis uses a variety of methods to study the creative process and correlates of creativity such as virtuosity and flow, using the Internet to broaden analysis to larger numbers of individuals while also shifting the focus to the personal creativity of less-recognised artists. These techniques include combinations of: large-scale, multi-user, longitudinal computer logging (Chapters 5, 7, and 8); a video case study, supported by interviews (Chapter 6); and both user questionnaires and psychometric-based user surveys (Chapter 9).²⁴

Amabile (1983) observed that creativity research had yet to reveal any effective ways to amplify creativity, but has instead highlighted many ways in which creativity is killed. Even more recently, Amabile (2006) noted that “creativity gets killed much more often than it gets supported” – and thus there remains plenty of scope to increase creativity by removing detrimental factors.

²³ Though Butler (2008) notes that the lack of an established canon of notated electronic music works may act as an impediment to the wider adoption of novel musical interfaces.

²⁴ Through summaries of composers’ historical accounts of the composition process (Graf, 1947; Boyd, 1992; Harvey, 1999; see 3.5 and Appendix B), this research can also be seen to draw on historic and biographical approaches to studying creativity (Plucker and Renzulli, 1999).

Rubin (1968) observes,

The research evidence unfortunately does not suggest that by using a prescribed scheme we can produce creativeness at will. What it suggests, rather, is that virtually everyone has more creativity than he makes use of, that different conditions flush it forth in different individuals, and that a given procedure tends to nurture a part, but not the whole of one's capacity.

controlling the environment

Creativity arises from interaction between a person and their environment (Csikszentmihalyi, 1996; Sternberg, 2003; with Lubart, 1995). Csikszentmihalyi (1996) observed that creative individuals actively adapt their environment to suit their styles and rhythms, and isolate themselves from the world. Consequently, strategies intended to support creativity – such as Nickerson (1999); Sternberg (2003); Amabile (1983, 2006) – do so by attenuating influences in the environment that discourage creative thinking. Moreover, these summaries of developmental and environmental factors affecting creativity have been, at least in part, borne out by empirical research, and consequently integrated into modern confluence theories of creativity, such as Amabile's *component model of creativity* (Amabile, 1983, 1996, 2006; see Section 3.3) or Sternberg and Lubart's *investment theory of creativity* (Sternberg, 2003; with Lubart, 1995).²⁵

strategies towards creativity

Many authors have suggested strategies concerning how the creative environment can be manipulated to improve support for creativity (Amabile, 1996, 2006; Nickerson, 1999; Plucker and Renzulli, 1999; Sternberg, 2003). Some recommendations are targeted at individuals, but others are framed as guidance for parents, managers or teachers, in developmental, business, or educational contexts. Many of these can be generalised to the individual (and user experience), but a few, such as those uniquely concerning social factors (e.g. *role-model creativity*, *teaching by example*), childhood or collaboration, lie beyond the scope of this research (see section 3.1) and are not detailed here.

In the following pages, Table 3 reviews their recommendations, in the context of four broader objectives, advocating support for EXPLORATION, EXPERTISE, MOTIVATION, and INDEPENDENCE.

²⁵ Investment theory is so named because it advocates that individuals, working in a social climate (see 3.1), should “buy low and sell high in the world of ideas [...] generate ideas that are relatively unpopular (buy low) and convince others of the worth of these ideas (sell high)” (Sternberg, 2003).

EXPLORATION	Sternberg (2003)	<ul style="list-style-type: none"> ● Encourage idea generation ● Tolerate ambiguity 	<ul style="list-style-type: none"> ● Redefine problems
	Nickerson (1999)	<ul style="list-style-type: none"> ● Stimulate and reward curiosity and exploration ● Provide opportunities for choice and discovery ● Use techniques / strategies for facilitating creative performance 	
	Amabile (1983) ¹ (2006) ²	<ul style="list-style-type: none"> ● Heuristics to generate novel ideas (e.g. counterintuitive approach)¹ ● Work style characterised by the ability to set aside problems¹ ● Freedom (concerning the process)² 	

Although new ideas form an intrinsic part of all creativity, individuals should be encouraged to explore multiple, alternative ideas, including those that might be considered counterintuitive, before committing to, or thinking too critically about, a specific approach. The environment must support provisional, incomplete, uncertain, and ambiguous expressions and solutions that can be easily changed or abandoned, without significant consequences.

EXPERTISE	Sternberg	<ul style="list-style-type: none"> ● Recognise that knowledge is double-edged sword 	
	Nickerson	<ul style="list-style-type: none"> ● Build basic skills ● Encourage acquisition of domain-specific knowledge ● Focus on mastery and self-competition 	
	Amabile	<ul style="list-style-type: none"> ● Cognitive style that involves coping with complexities¹ ● Challenge (that suitably stretches ability)² 	

Creativity requires domain knowledge and expertise (see section 3.6), together with specialist use of everyday skills; the acquisition and development of which must be supported and encouraged in the environment. The provision of a challenge commensurate with ability provides the opportunity to learn in a manner that is stimulating and rewarding. Mastery of tools and techniques helps tackle complexity, leading to higher attainment; but too much received knowledge encourages conformity.

MOTIVATION	Sternberg (2003)	<ul style="list-style-type: none"> ● Find what you love to do ● Accept delayed-gratification 	<ul style="list-style-type: none"> ● Tolerate mistakes
	Nickerson (1999)	<ul style="list-style-type: none"> ● Build motivation (especially intrinsic motivation) ● Use external motivation to reinforce intrinsic motivation ● Establish purpose and intention 	
	Amabile (1983) ¹ (2006) ²	<ul style="list-style-type: none"> ● Work style characterized by concentrated effort, an ability to set aside problems, and high energy¹ ● Enhance intrinsic motivation (by reducing extrinsic constraints)^{1,2} ● Challenge (that stimulates and satisfies)² 	

Creativity is best supported by intrinsic motivation (see section 3.3 and 3.7), which can be directly influenced by the environment. An environment must not interfere with an individual's will to be creative, and should promote activities that are inherently rewarding and fulfilling, such as the stimulation and satisfaction provided by an effortful challenge. Mistakes or failures should not be highlighted by external tests or evaluations, but tolerated and addressed without discouraging further activity. Less emphasis on external rewards also reduces the impact of delayed-gratification.

Table 3 – a summary of strategies developed to support an individual's creativity (continued on next page) by controlling environment factors, across four broad themes: Exploration, Expertise, Motivation, and Independence (based on Sternberg, 2003; Nickerson, 1999; and Amabile, 1983, 2006).

INDEPENDENCE

- Sternberg (2003)*
- **Identify and surmount obstacles**
 - **Question and analyse assumptions**
 - **Allow time for creative thinking**
 - **Take responsibility for both successes and failures**
 - **Maximize person-environment fit**
 - **Build self-efficacy**
 - **Take sensible risks**
- Nickerson (1999)*
- **Encourage confidence and willingness to take risks**
 - **Provide balance (between rule-following and rule-breaking)**
 - **Promote supportable beliefs (in one's potential; self-efficacy)**
 - **Develop self-management (meta-cognitive skills)**
- Amabile (1983)¹ (2006)²*
- **Cognitive style that involves breaking one's mental set¹**
 - **Freedom (autonomy)²**
 - **Resources (time, money and space)²**

The pursuit of novelty requires an environment that encourages uniqueness, independence, and autonomy. Self-efficacy and self-worth are required to enable an individual to take risks, persevere in the face of obstacles, and defy assumptions and established practices. External pressures (including time, money, correctness, recognition) must not be allowed to interfere with or limit creativity. Success or failure must be self-attributed, enabling an individual to identify and address their own strengths and weaknesses in private.

controlling the digital environment

Many of these strategies shift the individual's awareness away from the social factors of creativity, towards the activity itself and personal, psychological matters. Nickerson (1999) maintains that such an approach, especially concerning the reduced emphasis on the need for recognition (see Section 3.1), is critical to enhancing the creativity of the individual. Individuals, he argues, "need to believe that creativity is determined by motivation and effort to a significant degree"; not subject to some random, unpredictable extrinsic factor beyond their control.

Nickerson (1999) writes of the potential of computer software packages: "Can we assume that such tools – at least the best of them – will facilitate creativity?" He is optimistic, but noted a lack of existing research investigating the matter. More recently, HCI researchers have begun to consider the issues, strategies, challenges, and opportunities in digitally-mediated creativity (e.g. Resnick *et al*, 2005; Lubart, 2005; Knörig, 2006). Some of the findings of this emerging field are discussed in Chapter 4.

Moreover, many of the issues outlined in this section relate to the characteristics and requirements for "flow", a mental state characterised by challenge, absorption and intrinsic motivation that Csikszentmihalyi (1996) has explicitly linked with creative performance, and which has also been considered in the context of the computing, as discussed in Section 3.7.

3.5 creativity in music

*paucity of
composition
research*

Music, together with art in general, is a discipline that people implicitly associate with creativity (Sternberg, 1985). Yet, at the same time, creativity is a less-studied aspect of musical research (Sloboda, 1985/1999, 2005; D. Collins 2005, 2007).²⁶

A strong bias towards musical performance and musicology is also evident in modern teaching syllabi; and what studies of musical creativity there are tend towards studies of creative output, rather than process (Sloboda, 1985/1999). Sloboda (1999) notes a “general neglect of musical creativity in the arena of high art or 'classical' tradition which dominates schools, colleges, and universities.” Though there is a significant degree of creativity in performance, there remains a paucity of research on more overt acts of musical creativity, such as improvisation and composition (Sloboda, 1985/1999, 2005; D. Collins, 2005, 2007).²⁷

*composers and
psychologists*

Sloboda (2005) observes that the typically introverted nature of composers can create research problems, and also lies at odds with psychologists’ focus on the listener, providing the science of what the audience seeks, to facilitate its manufacture by composers.

*music as
architecture
or artefact*

Instead, he advocates studies of music as *architecture or artefact*, using a three-step strategy for practical psychological research concerning composition and musical creativity:

- (a) *determine the function(s) the [architecture / artefact] may perform*
- (b) *design structure that can serve that function*
- (c) *choose materials which will allow the structure to be made*²⁸

For example, music composition might best be supported by psychology research that suggests how notations can better capture composers’ musical creativity. With the increasing role of the computer in modern music production, this approach would seem to characterise the role of the interaction designer, who develops both the *system architecture* and *information artefacts* available to the computer-based musician, as well as the visual notations that form the user interface and define the interaction (e.g. Green and Petre, 1996).

²⁶ Harvey (1999) notes that his own work on musical creativity, in the 60’s, was actively discouraged by the music faculty of his university, which tended towards traditional, analytical musicology studies of the seventeenth-century.

²⁷ Sloboda (1985) noted, and later (1999) reiterated, “The reader should be warned that composition is the least studied and least well understood of all musical processes, and that there is no substantial psychological literature to review.”

²⁸ He also notes that “(b) and (c) often interact, in the sense that the structure can be to a certain extent determined by the available materials.” (Sloboda, 2005)

*composition as an
“ill-defined” problem*

Unlike improvisation, which can be seen as the timely solving of constraints (Johnson, 1980; Johnson-Laird, 1988; Alty, 1995; Thompson and Lehman, 2004), composition has been described as an ill-defined problem, with no pre-defined end-goal (Collins, 2007). Several authors describe composition as the transformation of this ill-defined problem into multiple well-structured problems, which can be solved separately (Sloboda, 1985; Collins, 2007), based on the identification and solution of constraints (Reitman, 1965; Lerdahl, 1988; Alty, 1995; Wiggins and Pearce, 2001). Once reduced, the composer can draw on a large, established canon of theory, technique, and practices in music. Reitman’s study of the composition process (Reitman, 1965), for example, focused on the fugue; a highly complex, yet highly systematised musical form, with many established rules and methods.^{29,30}

*finding, solving and
breaking constraints*

Johnson-Laird (1988) explores the issues of balancing freedom and constraint in creativity, arguing that the issue is not the degree to which creative processes are inherently computable or deterministic, but to which the individual *perceives* themselves to have choice and freedom of will. Constraints and aesthetic criteria are critical to the generation of ideas, but must exist in the creative individual as tacit knowledge, so as not to impact their sense of freedom. Conversely, conscious awareness of constraints benefits analytical thinking, but discourages creativity.³¹ Sloboda (1985) likewise notes that constraints that have so far been identified tend to apply to only “those events over which the composers have greatest conscious control”, and are thus confined to latter stages of the creative process (see Section 3.2).

Some authors argue that musical creativity, especially in composition, is about breaking rules rather than applying them (Auh, 2000). Burnard and Younger (2002) note that the degree of reliance on rules depends on the individual, who can balance the

²⁹ Indeed, the tractability of the fugue is evidenced by its link with improvisation (Mann, 1980), and has arguably led to the exhaustion of the form (Adorno, 1997). As such, even though Sloboda (1985) cites Reitman (1965) as one of the few observation studies of the composition process, one can argue that, like improvisation, the creativity exhibited is a special case. However, the methodology employed has since been applied to other composition activities (e.g. Collins, 2005, 2007).

³⁰ Based on research into formal grammars and psychoacoustics, Lerdahl (1992) goes as far as to propose a number of universal constraints on compositional systems. While he entreats readers to understand the constraints as psychological imperatives, the explicit aesthetic implications have been less well received (e.g. Boros, 1996). While his constraints identify the limits of musical complexity in order to be comprehensible to listeners, he advocates composers work as close to that limit as possible. He identifies Indian raga, Japanese koto, jazz, and most Western ‘art’ music as satisfying his criteria, but explicitly exempts both serialism and rock music, on the grounds of being respectively too far beyond or below this threshold. Whether one accepts these judgements or not, it is evident that such prescriptive systems lie at odds with many composer’s own motivations, philosophies, aesthetics and perceived artistic freedom (Sloboda, 2005).

³¹ Johnson-Laird (1988) offers this as an explanation for differences between critics and practitioners.

level of constraint and freedom to regulate the level of challenge and artistic independence (see Section 3.7). The implication for HCI is that research that identifies and reveals the constraints within a domain should not necessarily be used to formalise them in the interface, or explicitly bring them to the attention of the user. Rather, interfaces must be built around the development of tacit knowledge, and also afford the opportunity for users to discover and form their own perspectives (see 3.6 and 4.1.1).

the “blank canvas”

Many composers and researchers have observed that the most challenging and daunting part of the composition process is facing the initial “blank canvas”; establishing an initial musical idea (Graf, 1947; Boyd, 1992; Alty, 1995; Harvey, 1999; Deliege and Harvey, 2006; Collins, 2007). Once a theme has been laid down, composers can draw on the established devices of “transposition, augmentation, subdivision and recombination of elements, changes of rhythm, etc.” (Sloboda, 1985). However, it is these established, more formal processes that are the easiest to operationalise in the digital domain, contributing to a perception that related software (including DAWs and score editors) caters only for latter-stage productivity, offering only limited support for early-stage creativity (Blythe *et al*, 2007; Duignan, 2007).

Early stage musical creativity is characterised by unconscious thought processes that are hard to articulate, leading to a paucity of research (Sloboda, 1985), but which are most usually modelled using some derivative form of stage theory (see 3.2; e.g. Webster, 1990; Auh, 2000; Collins, 2005, 2007; Burnard and Younker, 2002; Burnard, 2007). Since a piece is most often the synthesis of multiple musical ideas (across time, melody, harmony, timbre, etc.) stage models are invariably adapted to accommodate multiple creative processes, operating iteratively (looped), recursively (nested), or in parallel. Creative threads can also be abandoned (ideas are discarded) or upended (ideas are revisited).³²

*exploring and
expanding*

Harvey (1999), reflecting on both his own experiences and those of other composers, observes that “unconscious inspiration” is not limited to just the inception of a musical piece, but also plays a significant role in revisions made in the latter evaluation stages of creativity (see also Deliege and Harvey, 2006). More broadly, Kratus (1989) divides the composition process into “exploration” and “development” phases, supported by divergent and convergent thinking styles respectively. Graf (1947) similarly described composition as a mix of conceiving, condensing and concentrating, expanding, elaborating and intensifying musical

³² A number of developments of stage theory in music are discussed in 3.5 and illustrated in Figure 2.

ideas.³³ Webster (1988) accordingly emphasises the importance of divergent thinking skills, such as musical “extensiveness” (ideation), flexibility, and originality, in addition to the subsequent application of convergent skills, such as musical syntax.

Whilst incubation is often associated with “time away” from the problem (Webster, 2002), waiting for the *intimation* of an idea (see section 3.2), composers like Stravinsky (Graf, 1947) and Harvey (1999) ascribe more focused effort (or “perspiration”, according to Harvey) to the process of exploring, expanding, and discarding different musical ideas. Boyd (1992) also describes professional music artists who see musical creativity simply as a “trial-and-error” process. Composers in the past have often turned to improvisation and experimentation with their instruments (especially the piano) to experiment with musical ideas, prior to notation (Graf, 1947). With the advent of the studio, musicians can record “jam sessions” for subsequent review (Boyd, 1992), a process that further shifts the emphasis from interaction with notation to performance.³⁴

sketching in music

Graf (1947) looks at the accounts of composers through history, and highlights the important role of sketching, which allows relatively cheap and noncommittal exploration of ‘fanciful’ ideas, and enabled composers like Beethoven to probe their unconscious and capture fleeting artistic moods. He describes the composition process as a gradual transition from unconscious to conscious thought processes, across several ‘moods’ characterised by a gradually decreasing playfulness and increasing commitment to specific ideas. The composer begins in a productive mood, playfully trying out musical fantasies, until the conception of a musical idea, which the composer then attempts to bring to form, aided by the informal, provisional format of the sketch, “until critical thinking alone puts the finishing touches to the tone figures.” Harvey (1999) also comments:

*Beethoven's sketch books are perhaps the most eloquent witness to the idea of inspiration as a gradual, 'clarifying' process: in them we can trace the emergence not only of themes but of entire structures, gradually becoming more and more crystalline.*³⁵

³³ Such descriptions compare with those articulated in the video study, in Chapter 6.

³⁴ This performance-based model of musical creativity is also evident in sequencers (see 2.1 and 4.2.3).

³⁵ Earlier composers, such as Mozart and Haydn, relied less on sketching, except as memory aids for larger, more complex works (Graf, 1947). However, the more formal rules of harmony, form, and structure in baroque and classical periods greatly facilitated the reduction and recollection of music (Harvey, 1999). Harvey (1999) also observes that these well-established rules and practices of the baroque period engendered less diversity, and greater conformity that ultimately impeded creativity (“achievement of synthesis”).



Figure 6 – Berio’s sketch for *Requies* (1983-5), illustrating corrections (❶), non-standard notation (❷), ink transcription from an earlier draft (❸), hurried additions (❹), side notes (secondary notation, ❺), and the lighter use of pencil in new work (❻). Image reproduced from Hall and Sallis (2004, p28).

The role of the sketch, as a tool for informally exploring ideas and thus a catalyst for creativity, is widely-recognised – not only in music (Graf, 1947; Hall and Sallis, 2004; Healey and Thiebaut, 2007 – see Figure 6), but also other areas of design (Blackwell *et al* 2008), including the design of user experiences themselves (Buxton, 2006). Schubert and Sallis (2004) state that sketches “are understood to be unfinished, open and provisional: the first unsure attempts to notate ideas, the significance of which is uncertain.” Sloboda (1985) describes them as, “necessary and enabling resources for the compositional process.” They allow composers to informally notate music without considering the legibility or acceptability of the idea to other musicians, allowing them to work quickly and without circumspection, at arbitrary levels of granularity or fidelity. The reduced investment of time and effort in producing a sketch also leads to fewer inhibitions to abandoning a path, should it prove unproductive. Moreover, paper sketches, made with pencil rather than pen, make it easy to make changes and further increase the provisionality of musical ideas – enabling artists to “avoid giving their original thoughts a permanent form.” (Graf, 1947)

Like sketching, digital notations must enable playful styles of interaction, to support both learning and creativity through exploration. However, these provisional, non-committal, and easily undoable (reversible) modes of manipulating a notation are not always available in music software (Duignan, 2007; Healey and Thiebaut, 2007). On the computer, data is stored and processed using volatile mediums that can be changed or erased much faster than an artist composer might rub out a pencil mark, but few user experiences are able to offer the opportunities and flexibility afforded by paper (Sellen and Harper, 2001). Blackwell *et al* (2008) observe a tension between the formalism required by computational interpretation and the informality desired in sketch tools, also noting limitations in physical control.

In music, Blackwell and Green (2000) used the *Cognitive Dimensions of Notations* framework (Green and Petre, 1996) to compare the computer use of musicians and programmers and found that “the musicians spent the majority of their time transcribing music from other sources, while the programmers spend more time exploring possible solutions.” Comparing the user experiences of each group, the study specifically highlights problems with *provisionality* and *viscosity* (a notation’s “resistance to change”) provided by the music editing environments.³⁶

In reviewing the limitation of computer music tools in supporting sketching, Healey and Thiebaut (2007) also talk about the need to support ‘vagueness’ and ‘ambiguity’. The cognitive dimension, *secondary notation*, is similarly related to a system’s provision for user inputs not adhering to the formalisms in the primary notation – that is, the opportunity to make freeform notes, annotations, etc. By definition, such notes cannot be interpreted (executed or played) by the computer, and serve only as an aid to user interaction, but one that may be useful in framing ideas before they are entered in an executable format. Section 4.2.4 further discusses this in terms of levels of liveness in the editing process.

Sloboda (1985) considers composition through an analogy with the musical creativity exhibited during improvisation:

The composer rejects solutions until he finds one which seems to be the best for his purposes. The improviser must accept the first solution that comes to hand... the crucial factor is the speed at which the stream of invention can be sustained, the availability of things to do which do not overtax the available resources.

³⁶ In Chapter 9, this analytical framework is used to further analyse tracker and sequencer styles of music software, with respect to these dimensions of the user experience.

*improvisation
and performance*

Improvisation depends on mastery of performance skill, such that control of the instrument is automatic, and the conscious mind free to consider the musical, rather than physical challenge – where fluid performances require dexterous fingering and fast motor control (see section 3.6). Alty (1995) sees improvisation as the realtime solution of a musical problem through the recognising and solving of well-learned heuristics or constraints, made possible by focusing the performer’s attention on a narrower window of time (Sloboda, 1985) and working within basic and largely preset musical structures, relating to tonality, harmonies, and form. Such “real time creativity”, argues Johnson-Laird (1988), is impossible without knowledge of constraints that hasten a solution.

*composition
and performance*

Composition, by comparison, involves the discovery of these constraints (Alty, 1995). The difference between improvisation and composition can thus be seen as a respective emphasis on problem solving vs. problem finding (see Section 3.2). The composer has broader latitude over both the music and time. The scope for complexity is increased, potentially raising the breadth or depth of the challenge, but the laxer timing constraint allows the challenge to be tackled at a more relaxed pace, reducing the requirements on real-time performance and improvising skill; reducing the anxiety without reducing the challenge. Production software based on realtime musical performance, such as DAWs or sequencers (see 2.1), thus do not exploit this opportunity to lower the threshold for musical creativity (see Scripp *et al*, 1988).

*controlling
musical time*

Burnard (2007) makes a phenomenological comparison of improvisation and composition, looking at the differences between the practitioners’ respective experiences. She characterises improvisation as time-constrained, task-constrained, and situation-driven; and composition as free, independent, self-driven and situation-owning. As with Sloboda (1985), improvisation restricts the opportunity for the personal exploration and appropriation of musical ideas and practice; in composition, the practitioner exerts controls over the situation, whereas in improvisation, they feel controlled. Sawyer (1995), who focuses on social perspectives of creativity in group-based improvisation, similarly makes a temporal distinction, describing improvisation as “synchronic” (immediate, single reception, ephemeral, where creative process and product are coincident) and composition as “diachronic” (delayed, multiple receptions, where the creative process is distinct from, but results in, the creative product).³⁷

³⁷ Collaborative systems rely on the presence of others as extrinsic sources of inspiration or motivation, contrasting the intrinsic motivation required for flow in personal creativity (see Section 3.1 and 3.7).

Thus, the flexibility and control of (musical) time becomes a critical factor in the UIs of composition software. Duignan (2007) notes that, while linear timelines are useful for building and finalising the overall arrangement of a piece, non-linear music systems allow a greater flexibility and provisionality that can facilitate creativity. He states, “By being able to arrange in real-time, producers can try out new ideas very rapidly and create results they would have otherwise overlooked”. He also notes that a “state of flux” can be maintained during music production, where artistic decisions are deferred later into creative process, in contrast to the premature commitment enforced by linear timelines (e.g. sequencers). Thus, beyond the linear, realtime requirements associated with performance and improvisation modes of studio-based musical production, tools for composers must be designed not just for the fluid control of musical variables (pitch, dynamics, etc.), but also fluid control of time itself (see Section 8.3) and support of an interactive composition process.

*studying
composition*

Beyond studies of musical output,³⁸ Sloboda (1985) identifies four ways by which one might inspect the composition process itself: composer reflections, studying sketches, observation (e.g. 'think aloud' studies), or by looking at improvisation instead.

*composer
reflections*

Graf (1947) attempts to collate the reflections of composers, regarding their creative processes, and place them in a coherent psychological context. Harvey (1999) can be seen as a similar anthology of perspectives and reflections that also encompasses modern composers of the 20th Century. His account lacks a psychological context, but is largely reconcilable with Graf's (Deliège and Harvey, 2006), demonstrating the earlier work's continuing relevance. More recently, Boyd (1992) provided an anthology of reflections from artists in popular music, framed by Maslow's theories of *self-actualisation*, creativity and motivation (Maslow, 1963, 1968; see Section 3.7). Such anthologies should be seen as invaluable resources for UI designers following user-centred approaches (see 3.4; e.g. Norman and Draper, 1986).

³⁸ Pioneering the *historiometric* approach to studying creativity, Simonton (1994) studied 15,618 classical pieces from the classical period, examining the first six notes of the main themes from the works of 479 composers, looking for trends and probabilities in their tonality and the divergences thereof, as indicators of the originality. The significance of such novelty (the “musical fame”) was then scored on a 32-point scale based on the piece's showing in various music dictionaries, combined with rankings by musicologists. Among other findings, he showed a correlation between creativity and productivity, arguing that composers have a constant probability of creative success, such that more prolific individuals are more likely to hit upon something that society will recognise as original. While Simonton concedes that the characterisation of the music he uses is simplistic (reducing works of several minutes or hours duration to only 6 pitches; and ignoring harmony, instrumentation, dynamics and other critical factors in music), but offers his studies as examples of the application of computer in studying creativity and musical composition.

studying sketches

Within musicological studies, Johnson (1980) notes a tendency to concentrate on the finished form of a work. The promise of sketches is that they come closer to showing the inception and evolution of a creative work, showing an author's consideration of different ideas and the chronology of their creative process.

Schubert and Sallis (2004), however, note several problems with studying the creative process through sketches, largely concerning the availability of work to study and establishing the context or meaning of surviving pages. Moreover, they can be seen as an example of the biographic approach to studying creativity and, like historiometric approaches (e.g. Simonton, 1994, 1999), rely on eminent, indisputable, singular instances of creativity (Plucker and Renzulli, 1999). Neither approach is readily able to correlate their findings with analyses of less successful (or even failed) attempts that have been discarded; nor are they able to explore any aspect of the creative process that leaves no material evidence.

*composer
observations*

A few studies have attempted to address this by observing the creative act in process, through longitudinal studies following a composer from musical idea to realisation (e.g. Reitman, 1965; Sloboda, 1985; Collins, 2005, 2007). While this approach still focuses on subjects with recognised professional experience and skills in composition, the threshold for recognition is lower than that demanded by historical record, and thus also promises broader insights and relevance concerning more commonplace creative practices. Several findings of Collins (2005, 2007) are discussed in relation to the video study, detailed in Chapter 6.

*longitudinal
case studies*

Collins (2007) identifies methodological challenges in case studies. The duration of a composition process is indeterminate, varying from an afternoon to several years, pursued solidly or intermittently over that period. Thus, it is difficult to ensure the presence of an independent observer, who may also disrupt or intrude on the activity. Studies often rely on self-reporting techniques, which depend on a practitioner's subjective reflections and awareness of their own thought processes (more recently supported by audio and video recording), and the semi-regular saving of MIDI files (Collins, 2005). Neither record allows the observer to probe a composer's interaction in detail, but audio and video tapes can identify questions that can be put to the artist later, in the form of structured discussions and interviews. Lastly, due to the work involved in observation, studies are usually limited to single subjects; the observer is thus responsible for mediating objectivity, selecting an appropriate subject for study, and drawing conclusions that can be generalised to a broader population.

*self-reporting and
self-consciousness*

Inevitably, an individual's reflections on their own actions and motives make them increasingly self-conscious, and can harm the creative process (Csikszentmihalyi, 1996; Nickerson, 1999). However, creative processes are most obviously disrupted by the recurrent need for the composer to interrupt their activity, to reflect and report on their actions. Moreover, artists are frequently distracted by the composing activity itself, and forget to make their reports. While this presents a methodological problem that affects the quality and completeness of the study (Sloboda, 1985; Collins, 2005), the oversight is itself significant, as it indicates that the composer has become so focused and absorbed in their work, they forget the outside world. Section 3.7 discusses this phenomenon in the context of "flow", a mental state of total absorption in an activity that has been associated with creativity (Csikszentmihalyi, 1996).

*creativity in
group composition*

Although collaborative creativity lies outside the scope of this research, social settings can also offer insight into individual creative processes. Nabavian and Bryan-Kinns (2006), for example, conducted a study of distributed cognition in group composition, in which the interactions and communications of the participants provides a commentary on the emergence of musical ideas. In line with stage-based theories of personal creativity and accounts of composer's working processes (Section 3.2; Figure 3.2), the study identified three contingent processes – *attainment* (assembling information), *experimentation* (idea generation and selection), and *structuring* (verification and finalisation).

*creativity in
music education*

Barrett (2005) advocates studies of composition in education (see Webster, 1989; Auh, 2000; Burnard and Younker, 2002, Hallam, 2002; Burnard, 2007), as a way to broaden the focus of studies of musical creativity; in which an emphasis on recognition is counter-productive, and where rewarding achievement needs to be balanced with rewarding effort. In this sense, the school environment can be seen to address or mitigate some of the limitations of other approaches: the restrictive and contrived creative scenarios in controlled experiments; the inviolable and unpredictable freedom of solo work in case studies; and the lack of access to the process or products of personal creativity in biographic and historiometric studies. Studies of computer-based learning environments are discussed in the next section, which attempts to identify the skills and techniques used in music, and composition specifically, as well as the respective opportunities and challenges afforded by the computer.

3.6 developing musical expertise

As established in the previous sections, expertise within a domain is widely recognised as an integral component of creativity (Amabile, 1983, 2006; Ward, Smith and Finke, 1999). However, a number of HCI researchers highlight a design bias towards the novice user, at the expense of more expert, experienced users (Gentner and Nielsen, 1996; van Dam, 1997). Paradiso and O’Modhrain (2003) have questioned “how deep a union research in musical controllers will be able to forge with the larger field of Human-Computer Interfaces, which generally emphasizes ease-of-use rather than improvement with long years of practice.” Hewett (2002) also notes that the expertise commonly associated with computers and technology focuses on finding efficient solutions to specific problems, rather than facilitating the more open exploration that is important in creative practice (see Section 3.1). This section explores what types of expertise technology must support to facilitate creativity in music composition.

*explicit knowledge
and tacit knowledge*

While formal music systems revolve around *explicit, declarative* knowledge (e.g. musical score, scales, rules of harmony), which is comparatively simple to convey in a UI, expert music interaction (including listening, performing, improvising and composing) depends on a considerable amount of *tacit, procedural* knowledge, which is far more difficult to articulate, teach, and often acquired through sensorimotor learning, for execution below the level of reflective consciousness (Dowling, 1999). McCullough (1996) argues that “software makers would do well to place more value on tacit knowledge: the best tools will account for levels of mastery and psychology of participation, and conversely tool users should get more leverage from software’s formal constructions.”

acquisition of skill

Fitts and Posner (1967) describe three stages of skill acquisition. In the initial *cognitive phase*, one executes a task consciously, reflecting on each step to gain an understanding of it. In the *associative phase*, repetitive practice then leads to the emergence of patterns in stimuli and actions, enabling one to prioritise stimuli by recognising their relative usefulness. In the final *autonomous phase*, the application of these patterns and priorities becomes increasingly automatic, enabling unconscious performance.

Developing the ability to automatically process tasks within a domain is a crucial step in developing the expertise required for creativity (Collins and Amabile, 1999; Weisberg, 1999; Boden, 2004). In composition, skill and technique does not increase the artist’s inspiration or creativity, but rather their ability to quickly and faithfully articulate creative impulses in notation (Graf, 1947;

Webster, 1987; Harvey, 1999). Boyd (1992), however, notes that some artists deliberately develop mastery of instruments to reduce the role of conscious mind, so that “the musician is more likely to tap into the unconscious mind.” Weisberg (1999) argues that unconscious, automatic processing, developed through prolonged immersion in a task, frees capacity that can be spent on the finding and recognising of novelty. Boden (2004) observes that domain-specific skill, such as that in music, is also a process of developing and specialising unconscious, everyday psychological abilities (such as noticing, remembering, and recognising) – until complex musical structures can be interpreted automatically.

the role of memory

In *skilled memory theory*, Chase and Ericsson (1981) describe how experts acquire encoding and retrieval skills in long-term memory that, with practice, increase access times to levels comparable to short-term memory, allowing for an effective increase in the capacity of their overall working memory. An extension of this idea, *long-term working memory (LTWM) theory* (Ericsson and Kintsch, 1995) also suggests how subjective knowledge-based associations work with similar memory mechanisms to provide musicians with individual understandings of music (Jänke, 2006). Alty (1995) describes a similar process in music, and argues that both short-term memory and long-term memory are important in composition, allowing composers to apply broad musical experiences to creative problems, such that “recall is far superior to recognition”.

The increased role and capacity of memory, in expert use, challenges the common precepts of usability design, which encourage the use of visual cues (“recognition, rather than recall”) specifically to “minimise the user memory load” (Nielsen, 1993). Whereas novices benefit from learning scaffolds, interaction that attracts attention to the individual steps taken harms the performance of experts (Beilock et al., 2002). Whilst experts can quickly recall commands or information from memory, the visual cues and searches guiding novices through a task support a much slower and more hesitant style of interaction (Gentner and Nielsen, 1996). Shneiderman and Plaisant (2005) note that the different densities of information favoured by novices and experts make it difficult to design scalable interfaces to suit both user classes. The implications and requirements of an increased role for memory in user interaction are further discussed in Section 4.1.1.

*composition skill
and editing scope*

In music, several studies of composition suggest that whilst novice composers focus on a narrow, local editing scope, working note-by-note, bar-by-bar; experts are also aware of larger-scale,

strategic and global factors (Davison and Welsh, 1988; Colley *et al*, 1992; Younker and Smith, 1996; Burnard, 2007), and use their knowledge of music to more efficiently chunk musical elements (Alty, 1995; Ginsborg, 2004). Chaffin and Lemieux (2004) define musical excellence as the ability to quickly switch between these low and high level (“Big Picture) perspectives, while maintaining concentration. Narrowing a learner’s focus to shorter musical passages, within the context of a longer piece, is an effective way of mediating the level of challenge and suggests a scalable way of then increasing it, as ability develops (Gabrielsson, 1999).³⁹

Accordingly, a scalable computer music authoring environment should support note-level, microscopic editing of shorter passages, to both simplify novice interaction and provide finer, detailed levels of control for experts, while also offering experts broader, macroscopic editing and song overviews. In computer music, Collins (2005) observed that, while expert composers dynamically alternate between low and high level editing perspectives in the sequencer, workflow and focus are not always maintained.^{40,41}

Smyth *et al* (1994) conclude “music requires many levels of representation, some of which are concerned with the knowledge of music itself, while others are auditory, spatial and motor.” Notably, they stress the importance of motor learning and control, and recommend three perspectives when investigating “flexible, well-learned skills”: the action as a physical operation, with physical and physiological constraints; movements as an operation in space, requiring a representation of such space; and the potential for the hierarchical structure of motor control to mirror structural meaning in the domain.

The important role of the hands has been critical to mankind’s evolution, by enabling the development of skills for completing complicated tasks (Wilson, 1998). Both computer and music interaction rely heavily on the accuracy, fidelity and dexterity of arm, hand and digit movement (Williamon, 2004), which are also central to the support of digital craft (McCullough, 1996).

³⁹ The importance of which is discussed in the context of “flow”, in Section 3.7.

⁴⁰ Collins (2005) also notes that most writing on the subject of composition concerns higher-level musical processes, even though a significant amount of the composer’s time is focused on finer, low-level detail. His study observed that such low-level edits were often characterised by absorbed, highly-focused interaction. This might indicate a flow state (see Section 3.7) and thus concern activity about which composers are less able to articulate in writing.

⁴¹ Sections 8.5 and 9.3 explore the advantages offered by DAWs that focus on shorter musical passages (patterns or loops), in comparison to more traditional sequencers based on linear timelines and project overviews (e.g. the arrange window).

Theories of motor control suggest that repetitive tasks can be learned and cued unconsciously. Such *neuromuscular facilitation* (or “muscle memory” – Chafe and O’Modhrian, 1996), has been used to explain expert use in activities such as touch-typing and piano-playing (Smyth *et al*, 1994). For example, in typing, exposure to common phrases (digraphs, trigraphs, etc.), and their respective sequences of physical actions, condition the motor and nervous systems to respond with little or no conscious reflection, freeing cognitive resources for application in the task domain.

spatial schemata

Smyth *et al* (1994) describe experiments that also demonstrate the use and development of generic spatial schemata for devices, such as the layout of the computer keyboard. Cohen *et al* (1990), for example, showed that the performance shown by touch-typists is not simply a product of specific well-learned motor sequences, but also of a generic knowledge of keyboard layout that enables them to maintain performance during unfamiliar sequences.

developing musical skill

Smyth *et al* (1994) observed similar mechanisms at work in piano-playing. Like the computer keyboard, the static, fixed layout of musical keyboard, enables the development of both spatial schemata and motor learning. Similarly, Thompson and Lehmann (2004) see both sight reading and improvisation as dependent on motor programs developed from exposure to a large base of musical knowledge and experience. Sloboda (1985) also makes a distinction between memorised instrument fingerings and ‘general knowledge’ developed by exposure to a wide-range of problems, where new fingering problems can be effortlessly solved at sight.

the role of experience

In a music teaching environment, a student is exposed to enough knowledge (i.e. declarative or by practical demonstration) to enable them to attempt a task, the execution of which allows them to develop the tacit knowledge from their own experiences (Sloboda, 1985; Boden, 2004). Mastery of a musical instrument is then further developed through regular, deliberate, and repetitive practice (Ericsson *et al*, 1993; Ericsson and Lehmann, 1996; Weisberg, 1999; Williamon, 2004).

the requirement of practice

Musical skill takes both considerable time and sustained effort to develop (Sloboda, 1985; Gabrielsson, 1999; Williamon, 2004). Virtuoso performance skill can demand up to 10 years (or 10,000 hours) of deliberate and disciplined practice (Ericsson *et al*, 1993),⁴² which “presupposes high motivation and extended effort,

⁴² A similar “10 years of silence” has been observed before even prodigious talents realise their first masterpiece (Hayes, 1989; also cited in Weisberg, 1999; and Chaffrin and Limieux, 2004). For example, Mozart’s remarkable childhood concertos and symphonies are generally seen as studies or imitations of other composers, rather than original works in their own right (Weisberg, 1999).

learning by ear

full attention during practice” (Gabrielsson, 1999). Necessarily, the development process must itself become a source of motivation; as progress is made and attributed to the effort, self-efficacy and self-perception increase, spurring further effort (Candy and Edmonds, 2004; Chaffin and Limieux, 2004; see 3.7).

In the last century, a number of music pedagogies (notably: the *Suzuki Method*, *Dalcroze Eurhythmics*, *Kodály Method*, *Orff Schulwerk*, and *Music Learning Theory* – Shehan, 1986; Gordon, 1997) have emerged that defer or eschew the explicit learning of theory or notation until after students have a tacit understanding of musical structure, developed from extensive exposure to music, often in combination with singing and movement. These *learning by ear* approaches attempt to mirror the way children learn languages through listening, imitation, and experimentation, while also emphasising the role of the body and motor skill in music (Kreitman, 1998). Distinct from literacy, the student implicitly identifies structures and patterns in sounds and actions, such that enable predictive and generative interaction and mental simulation of music. In *Music Learning Theory*, Gordon (1997) calls this “audiation”,⁴³ arguing that it forms the foundation, and provides the musical context, for subsequent developing notational literacy, as well as performing and composition skill. A similar approach to learning music and composition is observed in tracker interaction, in Chapter 6 and later chapters.⁴⁴

formal music education

However, Webster (1989) observes of music education, “More often than not, we tend to teach our art only by rule or by rote”. Students acquire, and become entrenched in, an understanding of music based on theory and polished performances of set works, encouraging conformity and correctness, inhibiting creativity (Sloboda, 1985; Webster, 1989; Harvey, 1999).^{45,46} Barrett (2005, 2006) also observes the inhibitive influence of classical training on the creativity and motivation of younger musicians; during training, performing artists are exposed (and ultimately disposed) to techniques that encourage common practice, rather than novel and independent ways of thinking. Imitation and exposure to a large repertoire of music is an inherent component of both

⁴³ Audiation is the process of hearing music in your head, in the absence of physical sound as a stimulus – the mind’s ear – and is musically analogous to how individuals can think in terms of language, without reading, listening or speaking. (Gordon, 1990)

⁴⁴ As enabled, for example, by rapid musical feedback after tinkering with the notation (see Chapter 8).

⁴⁵ See also Alty (1995); Weisberg, (1999).

⁴⁶ Such expertise also reduces the chance of serendipitous ‘mistakes’ that lead creative individuals down paths they would not otherwise have considered (Alty, 1995). McLean (2011) even observes that the *error proneness* of a music notation or interface might not be as undesirable as in other task domains, or usability practices.

performance and improvisation tuition (Thompson and Lehmann, 2004),⁴⁷ but also a deterrent to creative growth (Boyd, 1992; Simonton, 1999). While exposure to the works in a domain can inspire artists to innovate, appropriate, and combine the styles of others (Alty, 1995; Csikszentmihalyi, 1996; Harvey, 1999; Weisberg, 1999), too much knowledge can encourage convention rather than invention (Sternberg, 2003; Feinstein, 2011).

The opportunity for creativity itself can also be an important, if underexploited, motivation to develop expertise (Torrance, 1962; Sloboda, 1985; Collins and Amabile, 1999; see section 3.3). Swanwick and Tillman (1986) observe that, whereas there is a necessary sequence to developing musical skill (in which creativity relies on an ability to express ideas, enabled by sufficient mastery of a tool), there are opportunities to introduce personal, playful, self-motivated and creative activities throughout development. Kratus (1989) also observes that current musical pedagogies are based primarily on performance and listening, and identifies a benefit to supplementing them with creative activities, such as improvisation and composition, much earlier in development. His studies noted that young children were not only already able to compose music with meaning, but did so with considerable enthusiasm. He also noted that older children increasingly moved away from a focus on generating new ideas, towards a *product-oriented* approach based on the development and refinement of fewer ideas (Kratus, 1989).

The *Orff Schulwerk* is one of the few pedagogical approaches to actively promote musical creativity from the outset (Shehan, 1986) – enabled by the central role of play, whereby “the materials used in all areas should be simple, basic, natural, and close to the child’s world of thought and fantasy.” (Shamrock, 1986) Students begin with *exploration* (of the relationship between sound and movements), acquire basic rhythmic and melodic performance skills through *imitation*, where they learn to recognise patterns that ultimately enable unprepared *improvisation* of new patterns in realtime group activities; all culminating in *composition* (or “creation”), where material from previous phases

⁴⁷ Thompson and Lehmann (2004) also observe that improvisation is rarely taught. Along with composition, Johnson (1980) sees this as a consequence of a wider perception, in traditional music, that creativity cannot be taught. More generally, the concentration on performance technique, music theory and musicology in musical research and curricula might be seen as a consequence of the practical limitations of didactic (factual, critical, or theoretical) teaching, which result in the marginalisation of predominantly autodidactic (self-taught) musical creativity, such as improvisation and composition. Performance, which similarly relies on implicit and procedural knowledge, only remains in the musical syllabus because it is seen as the common-denominator of all musical activities – expertise that enables improvisation, and thus musical creativity and composition.

is combined to prepare works in simple musical forms, based on literary material (poems or stories). Inspired by early medieval music, set pieces draw heavily on simple (but varying) rhythms, modal scales and *ostinati* (short repeated phrases), which greatly simplifying the learning of pitch, tonality, and melody.⁴⁸ The use of simpler musical building blocks makes the domain easier to master, lowering the threshold for creativity, but still providing a scalable challenge as different elements are combined to engender more complexity. This use of “simple primitives” is further explored, in an HCI context, in section 4.1.1.

In her study of popular composers and songwriters, Boyd (1992) observes, “Many artists resisted the limitations imposed by formal music or art lessons as... feeling the need to break free of all limitations.” She notes that many show an independent and rebellious attitude in developing technique and personal style. According to Sloboda (1985), “Idiosyncrasies of self-teaching can be advantageous, in comparison to the rigorous formal training, often to the point where individuality becomes submerged.”

Drawing on the developmental epistemology of Jean Piaget, Knörig (2006) notes that discovering a musical concept for oneself can lead to a greater, more flexible understanding, compared to what might be formally imparted. As in the *Orff Schulwerk*, such personal exploration and experimentation as a learning strategy encourages creative thinking from an earlier stage of development. Moreover, these intrinsically-rewarding activities instil self-efficacy and a feeling of autonomy, also benefiting creativity (Amabile, 1983; Nickerson, 1999; see Section 3.7).⁴⁹

Scripp *et al* (1988) explored uses of the computer that allowed musically-untrained adults to tackle complex music composition tasks, by using playback to guide their interaction, rather than interpreting their music through visual notation or performance:

Students using computer software to solve their counterpoint or harmony homework appear more likely to take advantage of the editing, revising, and playback functions of the computer without being distracted by the demands of musical performance beyond their level of proficiency. Musical composition can be more objectively related to its notation through computer playback, a noninterpretive rendering of the score.

⁴⁸ This effectively reduces the octave from twelve to seven pitches (e.g. the white notes on a piano), obviating the need for accidentals (flats and sharps) or knowledge of key.

⁴⁹ Indeed, in this sense, it can be seen as a P-creative act in itself, whereby the individual perceives newly discovered concepts as both novel and useful (Grüber and Wallace, 1999).

Their study is also significant in that it illustrates a capacity for musical creativity in adults who have not been subject to extensive musical training from an early age. Rather, a degree of audiation skill develops naturally in most individuals, through years of music listening, enabling composition through the “interactive exploration” of a digital notation. The findings of Gall and Breeze (2005) – who note a similar democratisation of musical creativity in computer-based composition, afforded by a focus on musical feedback – also suggest a less conscious, more synthetic style of interaction arises. Chapter 8 observes a similar interaction style and learning process in tracker interaction.

*embodied interaction
and music cognition*

Knörig (2006) advocates embodied learning approaches in digitally-mediated musical creativity. Following the earlier arguments of Winograd and Flores (1986) and Dourish (2001), and based on Heidegger’s phenomenological distinction between tools that are *zuhanden* (ready-to-hand) rather than *vorhanden* (present-at-hand), he looks at *tangible user interfaces*, for ways the body can “extend itself through external devices.”

Leman (2008) makes a similar case for music performance technologies to exploit and support the development of embodied music cognition, through the provision of rapid *action-reaction cycles* that enable motor learning by assuring perceptible relationships between actions and objects, or cause and effect. He argues that musical experiences mediated by notations based on “abstraction, conceptualisation, and verbalisation” contribute to “indirect involvement” in music, and seeks ways of using gesture-based descriptions of music (e.g. movement) to reconcile semantic, linguistic-based descriptions (e.g. emotions) and sensory, signal-based descriptions (e.g. sound, waveform).⁵⁰ Rollo May (1975) expressed similar concerns; that “technology [can] serve as a buffer between us and nature, a block between us and the deeper dimensions of experience.” Indeed, Boyd (1992) quotes May to describe a common perception of technology among musicians, particular in the studio, that technology can remove the spontaneity, “feel” and touch from music making.

*formalism
and metaphor
in the GUI*

Graphic user interfaces in music software are often based on formalisms, notations, theory, and visual metaphor to standard practices in traditional, professional, and studio-based music production (Duignan, 2007). The emphasis on visually-mediated tasks through these notations, rather than the tightly-coupled motor actions and sound responses inherent in live music

⁵⁰ Indeed, this research can be seen as an adaptation of some of Leman’s concepts, when applied to necessarily notation-based musical activities, such as composition (see Chapter 3).

interaction, is unlikely to support the learning experiences advocated above. As a learning mechanism, such use of metaphor facilitates knowledge transfer, rather than its development (Blackwell, 2006; Venkatesh, 2007). At the same time, the increased standardisation reduces the opportunity for creative self-expression, and users become less able to appropriate the tools they use for their craft (Kitzmann, 2003; Blackwell *et al*, 2008).⁵¹

Gentner and Nielsen (1996) identify problems for expert interaction, in the WIMP and GUI-based approaches to usability, which entail “a trade-off between ease of learning on one hand, and ease of use, power, and flexibility on the other hand.” Blackwell (2006) specifically describes their “Anti-Mac” philosophy as an attempt to shift the use of metaphor in UI design towards a less deterministic, less structural approach that would enable creative interpretation and freedom. Creativity research has also established the need to allow individuals to develop their own metaphors, in their perceptions of a creative domain (Nickerson, 1999; in music, Webster, 1989).

towards computer-aided composition

Whereas production tools like sequencers and DAWs draw heavily on previously learnt musicianship,⁵² developing computer-based composition software that supports digital creativity requires the design of user experiences that support intrinsically-rewarding learning processes, based on exploration, discovery, and development of musical concepts situated within the digital music environment itself – in input devices that support motor learning, feedback mechanisms that allow learning by ear, and visual notations that support experimentation and scalable levels of musical complexity.

3.7 motivation and flow

Previous sections established the critical role of motivation, in the pursuit of creativity (3.3) and development of expertise (3.6). This section explores the various roles and manifestations of motivation in both personal and digital creativity, looking at the implications for the design of the user interfaces and experiences. It aims to highlight an implicit contrast between the sources of motivation required to support virtuosity in a user interface, and those that characterise the more conventional pursuit of usability.

⁵¹ Shneiderman (2002) reasons that computers are best suited to *evolutionary* rather than *revolutionary* creativity because of the inflexible paradigms in software that can “restrict your thinking”.

⁵² For example, DAWs require knowledge of the electronic studio, sequencers require performance skill, and score editors require notational literacy.

Amabile (1983, 1996, 2006, with Collins, 1999) distinguishes between *intrinsic motivation*, where a task is its own reward, and *extrinsic motivation*, where a task is undertaken for some external incentive (e.g. salary, prize, recognition, duty, fear). Creativity, she argues, depends on intrinsic motivation, whereas extrinsic factors can impede it (see also Crutchfield, 1962; Hennessey, 1989; Csikszentmihalyi, 1990, 2006; Nickerson, 1999; Plucker and Renzulli, 1999; Runco and Sakamoto, 1999).

In intrinsic motivation, individuals focus on, and immerse themselves in, the challenge, process, or task itself; in extrinsic motivation individuals focus on the end-goal, product, or reward (Crutchfield, 1962; Lubart and Sternberg, 1995; Nickerson, 1999). Extrinsic motivation can thus distract attention from a task (Crutchfield, 1962; Amabile, 1983; Collins and Amabile, 1999), and undermine intrinsic motivation (Plucker and Renzulli, 1999). Notably, extrinsic factors can increase the involvement of ego in a task, make the individual self-conscious, and introduce the fear of failure or rejection – discouraging risk-taking and experimenting with new ideas and encouraging conformity (Crutchfield, 1962; Nickerson, 1999).

Some forms of extrinsic motivation can be useful, either in the absence of, or support of, intrinsic motivation. Though sustained participation in a domain requires intrinsic motivation (Amabile, 2006; in music, Chaffrin and Lemieux, 2004), encouragement from peers, parents, or teachers can be important in seeding initial interest (Crutchfield, 1962; Collins and Amabile, 1999; Plucker and Renzulli, 1999) and providing positive, confidence-boosting or constructive feedback that leads to independent thinking or alternative perspectives during early development (Moran and Liou, 1982; Runco and Sakamoto, 1999). Amabile (1996) describes these extrinsic motivations as *synergistic*, encouraging a sense of control; rather than *non-synergistic*, encouraging a feeling of being controlled. Without the controlling influence, individuals are more disposed to the unconscious, playful styles of thinking that favour creativity (Koestler, 1964; Collins and Amabile, 1999).

Maslow (1963) asserts that beyond our basic requirements (e.g. health, security, love), *self-actualisation* (the realisation of one's potential) constitutes the pinnacle of man's "hierarchy of needs", and provides the drive for creativity (see also Boyd, 1992⁵³; Collins and Amabile, 1999; Sternberg and Lubart, 1999; Knörig, 2006).⁵⁴

⁵³ Boyd (1992) also notes similar philosophies in the work of psychiatrists Carl Jung and Rollo May.

*self-efficacy
and challenge*

As an individual builds experience, they develop *self-efficacy*; the confidence in one's abilities to set and attain their own goals, instilling the courage to pursue new paths and challenge conventions, required for development and self-expression in a creative domain (Boyd, 1992; Amabile, 2006). Creative individuals are intrinsically motivated by challenging tasks that stretch and extend their abilities and creative power (Torrance, 1962; Collins and Amabile, 1999). A similar cycle exists in music, where practice leads to expertise, motivating further practice (Hallam, 2002), and where experts revel in their perceived creative power (Sloboda, 1985; Candy and Edmonds, 2004).

*self-attribution
and effort*

In personal creativity and development, *self-attribution* is implicit; credit and blame are not diluted by external influences (Chaffin and Lemieux, 2004). While success motivates an individual to try harder challenges, failure demands that they acknowledge their weaknesses (Csikszentmihalyi, 2006). External influences can allow an individual to dismiss criticism as unfair or attribute the failure to the teacher, environment, or other factors beyond their control. Software interfaces play a similar role, as Magnusson and Mendieta (2007) observe,

People see it as their fault if they cannot play the instrument properly, not the imperfection of the instrument design itself. This is different with digital instruments [...] where people are more likely to criticise and see the limitations as weakness of the design rather than their own work methods or understanding of the system.

In the study, musicians cited “direct and natural” interaction as a key property of acoustic instruments, in contrast to digital music experiences, which were considered “disembodied”.⁵⁴

Computing approaches that automate or abstract complex processes not only create “black boxes” that impede a user's understanding and control of a system, restricting their creative freedom, but can also remove the challenges and efforts required to create an intrinsically-rewarding user experience (Resnick *et al*, 2005). McCullough (1996) argues, “Our use of computers ought not be so much for automating tasks as abstracting craft.”

Similarly, Ryan (1991), remarks “though the principle of effortlessness may guide good word processor design, it may have no comparable utility in the design of a musical instrument. In

⁵⁴ Jordan (2002) also uses this hierarchy to advocate pleasurable products, based on intrinsic motivation, as the next step for HCI beyond usability.

⁵⁵ Similar to Leman's contention that notations can engender “indirect involvement” in music (see 3.6).

designing a new instrument it might be just as interesting to make control as difficult as possible.” Linson (2011) also raises Norman’s call for “appropriately complex” interfaces (Norman, 1993), arguing that musicians tolerate and embrace greater complexity than that assumed by many usability approaches.

The role of external evaluation has also been cited for its negative effect on creativity (Rogers, 1954; Crutchfield, 1962; Amabile, 1983; Hennessey, 1989; Runco and Sakamoto, 1999; Plucker and Renzulli, 1999). Nickerson (1999) observes, “Fear of failure, fear of exposing one’s limitations, and fear of ridicule are powerful deterrents to creative thinking.” By contrast, the free, personal environment is inherently forgiving of failure; *self-evaluation* is inherently biased towards a positive outcome, and able to downplay a negative one (Hallam, 2002). The failure is still evident to the individual, but never becomes the subject of ridicule, and so can be addressed privately without harming *self-worth* (Collins and Amabile, 1999).

Recognising this, composers use the private nature of sketches, which do not have to be perfect or even legible to others, to overcome their inhibitions and explore original, incomplete and imperfect ideas (Graf, 1947; see 3.5). This affordance is less well supported in digital notations, where not only must music remain communicable to the computer (or synthesizer, etc.), but where rigid interfaces become unwieldy when users stray from their intended use. Collins (2007), for example, observed that sequencer users must frequently stop and “tidy” the UI in order to progress.⁵⁶

As a design principle, Resnick *et al* (2005) argue that creative support tools must support collaboration, and exploit the connectivity offered by technologies such as the Internet. The suitability of this should be carefully considered in artistic endeavours, such as music. Bryan-Kinns and Hamilton (2009), for example, explore the design of user experiences that support *mutual engagement* (or *group flow* – Csikszentmihalyi, 1996) in music. In their study, they identify problems with “lack of control” and “clash of ideas”, attributed to concerns over intellectual ownership and social awkwardness. Framed in terms of motivation, these problems can be explained by the role of ego and perception of self, whereby events in the activity that more explicitly draw attention away from the task and towards other

⁵⁶ The visual aesthetic of the interface can also imply formalism (Blackwell *et al*, 2008). In music, a tendency towards neatness and correctness is implicit in the typeset, print-quality notations used by score editors, which mimic that of a final manuscript rather than a hastily-pencilled sketch.

people increase one's self-awareness, impeding flow.⁵⁷ Whilst Sawyer (2006) observes that group flow can help individuals attain their own flow state, this presupposes that synergy within the group is strong, such that members feel confident (e.g. with their ability) and can interact naturally without being self-conscious. Boyd (1992), for example, highlights the challenges of striking the right "chemistry" when musicians collaborate, and a tendency for song-writers and composers to work, or seek time, alone and isolated from the world (Graf, 1947; Getzels and Csikszentmihalyi, 1976).

Fencott and Bryan-Kinns (2010) noted that digital collaborators make extensive use of private working areas to develop musical contributions before introducing them to the group. Largely as a result of technological limitations, tracking practice similarly revolves around writing music in isolation, and sharing it with others upon completion; enabling composers to selectively engage with the community, and only when they were satisfied with their efforts and confident of social acceptance.⁵⁸

*motivation from
online communities*

A large number of online communities (e.g. the *demoscene*, see 2.2.2) cater for varying musical tastes and skill levels, and provide a source of extrinsic motivation that is often synergistic, where technological limitations create a detachment that can insulate the individual. On a basic level, simply the increased opportunities to display creativity can help motivate individuals (Csikszentmihalyi, 1996). Cook (2009) also found that online communities tend towards positive, constructive ("reinforcing") feedback, refraining from disparaging or negative feedback. Flexible, ambiguous, and possibly-inflated perceptions of audience may also motivate initial attempts at social creativity. Similarly, the relative anonymity can remove inhibitions (Junglas and Steel, 2003), giving individuals the confidence to expose their work or seek help.

Like the presence of other people, a UI must be considered for its potential to expose external influences in the user's creative process. Lubart (2005)'s anthropomorphising of the role of the computer as a "partner" in the creative process should perhaps be considered for its impact on motivation.⁵⁹ Moreover, aesthetics,

⁵⁷ This fits with Bryan-Kinns *et al* (2007)'s discovery that engagement improves when the identity of others is hidden, and thrives in the absence of explicit interaction, such as verbal communication, between participants.

⁵⁸ The *Renoise* tracker's default full-screen, self-contained, *DirectX* environment implicitly retains this impression, separating the user from other OS processes and distractions.

⁵⁹ Lubart (2005) sees four roles for computers in creativity: the *nanny*, which actively intervenes to enforce deadlines, prompts breaks, and handles housekeeping tasks; the *pen-pal*, which enables the

*the computer as
creative environment*

conventions, practices, or assumptions about a domain, implicit in the design of software, but in conflict with the user's values, may also be perceived as an external influence or controlling factor.

In this research, the role of the computer is thus defined by analogy to the *creative environment* (see also section 3.4); an integrated container for tools and processes that support free and personal exploratory creativity within a virtualised domain that can be appropriated by the user.⁶⁰ Many strategies for enhancing creativity (see section 3.4) focus on enhancing an environment's capacity to support intrinsic motivation (Nickerson, 1999; Hallam, 2002; Sternberg, 2003), and can thus also be adapted to the user interface (e.g. Shneiderman et al, 2005).

More literally the virtualisation of a creative environment, the *desktop studio* draws heavily on visual metaphor to simulate the electronic recording process (Duignan, 2007; with Biddle, 2005), but which also imposes preconceptions of music and the production process that depend on performance skill and external devices, rather than facilitating digitally-mediated creative exploration (see Section 2.1).⁶¹ Boyd (1992) records musicians' mixed feelings towards the creative affordances of the studio, but notes that its "timeless, womblike atmosphere" can be conducive to focusing and immersing oneself in the musical activity. Graf (1947), Boyd (1992) and Harvey (1999) also describe how composers and songwriters control their environment to cut off the outside world, helping them become more absorbed in the music.

immersion and play

Intrinsic motivation is often characterised by deep involvement and immersion in an activity (Crutchfield, 1962; Policastro and Gardner, 1999), in which all attention, awareness, and cognitive ability is focused on the task itself, without regard to external factors or goals (Golann, 1962; Collins and Amabile, 1999). For example, while Ericsson and Lehman (1993) argue that the effort involved in developing musical expertise is not inherently enjoyable, Boyd (1992) observes that when musicians become immersed in music, through their instrument, "the distinction between work and play [becomes] shadowy."

communication of ideas to other users; the *coach*, which offers alternative perspectives and analogs to "jump-start" the process; and the *colleague*, which collaborates with the user on the problem.

⁶⁰ Unifying analogies in video gaming, computer security, and software testing, we might look to design the creative environment as a "sandbox" – enabling playful, open exploration of a virtualised domain; moderating the social presence, to protect the creative individual from outside intrusion or enable them to engage with it on their own terms; and creating a safe environment for learning and experimentation, where practitioners do not have to worry about the consequences of their acts or ideas. Some of these concepts have already been applied to digital learning environments (Johnson *et al*, 2005; Bellotti *et al*, 2009).

⁶¹ See also *FL Studio*'s more integrated, software-based studio, discussed in Section 8.5.

Similar characteristics are found in the leisurely act of playing. When an activity becomes more playful, individuals will explore possibilities beyond the prescribed bounds of a task and become disposed to creativity and learning; tapping into imagination, fantasy, curiosity, energy, and whimsy (Nickerson, 1999; see also section 3.5). The link between immersion, play, and intrinsic motivation is evident in video games (Sweetser and Wyeth, 2005; Jennett *et al*, 2008; e.g. virtual reality, role-playing), which have also been adapted to provide environments for learning and creativity, in the form of “Serious Games” (Michael and Chen, 2005; Bellotti *et al*, 2009).

personal computing

The hardware of the personal computer (screen, keyboard, and mouse) has been criticised for its “absorbing” physical presence, and tendency to isolate the user from the outside world (Knörig, 2006; Armstrong, 2006; see also Weiser, 2001). Knörig (2006) describes the computer “not as a tool, but as an own world”. From the perspective of motivation, this isolation may serve to mask extrinsic factors in the environment, while the screen acts as a focal point for the user’s attention. However, this also increases the onus on the software interface to maintain the user’s focus and limit other references to agents, objects, and processes outside the user’s creative activity, which can be difficult in modern connected, multi-tasking desktop environments.⁶² Indeed, this can be seen as an implicit advantage of early music and tracker programs, which ran as full-screen applications in single-task environments (e.g. *Amiga*, *DOS*; see Section 2.2). The thin OS layer and low-level hardware integration enabled such programs to appropriate and adapt the computer, providing more of a dedicated interface for music, closer to that found in embedded devices like hardware samplers or sequencers.

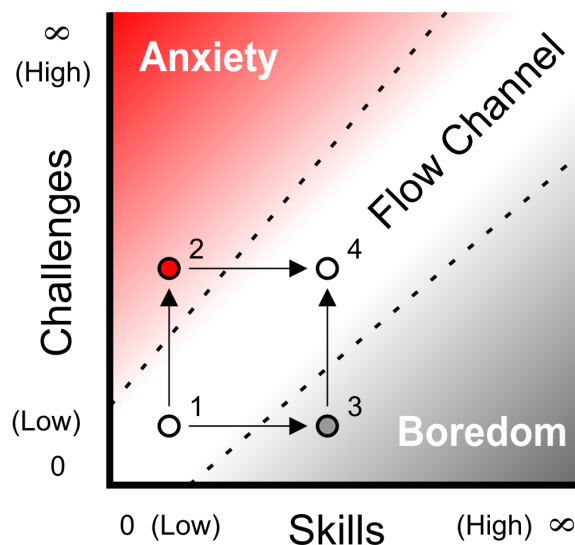
In the user experience, immersion is also supported by software that focuses the user’s attention on the music, rather than the interface (Leman, 2008). This can be achieved by emphasising musical feedback over visual; supporting non-visual interaction through memory and sensorimotor skills (see Section 3.6); enabling playful exploration of the domain (e.g. sketching; see Section 3.5); allowing users to appropriate the affordances of objects within a UI to support their own working styles and understandings of music; and otherwise minimising outside distractions or controlling influences. Specific design implications of these strategies are explored in Chapter 4.

⁶² Section 8.4 highlights problems when layered, window-based software environments not only divide a user’s attention and require extra management of the workspace, but also create a hidden background.

Figure 7
model of the flow state
 in which challenges and skills are balanced to regulate boredom and anxiety.

Examples

Novice individuals can experience flow in simple tasks (1), but increasing challenges can lead to anxiety (2), which is met by increasing skill (4). Conversely, a rise in skill must be met by rise in the challenge, so as to avoid boredom (3).



Csikszentmihalyi's
 "flow" theory

"Flow" theory (Csikszentmihalyi, 1990, 1996, 2000) provides a useful framework that brings together the themes discussed in this section: the individual, focus, immersion, intrinsic motivation, challenge, skill, and creativity.⁶³ It describes a mental state, where a delicate *balance of challenge and ability* leads to a *feeling of control* and a *loss of self-consciousness*, engendering a working environment that can benefit creativity (Csikszentmihalyi, 1996). As Figure 7 illustrates: too much challenge and an individual becomes anxious, too little and they become bored. Over time, ability increases, requiring greater challenges to maintain flow, ultimately leading to the development of mastery in a domain. In this context, the flow concept describes an intrinsically rewarding path to building ability, through enjoyable and fulfilling challenges, matched to the individual. Table 4 lists the nine components that often characterise a flow experience.

flow in
 computer use

Norman (1993) proposes flow as a basis for introducing informal learning, challenge, and reward into the user experience. Several researchers have already drawn upon flow theory to investigate the enjoyment and learning afforded by the challenge of video games (Jones, 1998; Sweetser and Wyeth, 2005). Church, Nash and Blackwell (2010), based in part on the research in this thesis, explore flow in notation uses in both music and computer programming. Originally delivered as a keynote on flow in programming and HCI,⁶⁴ Bederson (2004) notably emphasises the

⁶³ Flow theory, as "the psychology of optimal experience", can be seen as a development of Maslow's "peak experience", which Boyd (1992) uses in her interviews with modern composers and songwriters.

⁶⁴ *Human-Centric Computing 2002* (now *Visual Languages and Human-Centric Computing, VL/HCC*).

Clear Goals

In artistic creativity, knowing end-goals can be difficult or even counterproductive, but artists should have (possibly unconscious) knowledge of actions to perform to make progress, at any given moment.

Direct and Immediate Feedback

Actions must provoke an immediate response, to allow individuals to assess their progress, and adapt to problems. Direct feedback on individual actions enables finer control and focus, but feedback on overall progress enables sustained concentration and can provide an autotelic reward.

Balance of Challenge and Ability

An activity must challenge an individual to provide an intrinsic reward, but too difficult a task that exposes insufficient ability can make them self-conscious (see below). Specifically, a task should stretch the individual just beyond their ability, leading to increased skill over time (Amabile, 2006).

Action-Awareness Merging

The task domain should be the limit of the individual's awareness, immersing them in the activity, so that all attention and skill can be applied to meeting the challenge presented.

Concentration and Focus

Flow activities are often characterised by a momentum or continuity of action that demands sustained concentration on the immediate task at hand, uninhibited by outside distractions.

Sense of Personal Control

The individual has an implicit confidence to meet the challenges exhibited in a task, and total control within the task environment. There is no worry of failure, or reflection on consequences.

Loss of Self-Consciousness

Undivided focus on the activity removes the doubt prompted by an individual's tendency to monitor their appearance to others, preventing them from acting or trying new approaches. Ego becomes irrelevant; though subsequent reflection, following success, ultimately leads to improved self-image.

Distorted Sense of Time

The subjective experience of focusing continually on the present and exclusively on the world of the task (see above) can detach an individual from their perception of time. Hours may seem like minutes; but great ability that affords fast thinking can also seem to slow down fleeting, complex moments.

Activity becomes Autotelic

Under the above conditions, the activity can become intrinsically-rewarding. Initial involvement may be exotelic, requiring an extrinsic motivation, but increasing ability brings rewards, and instil self-efficacy (Maslow, 1968), that enable the task to be pursued and enjoyed for its own sake.

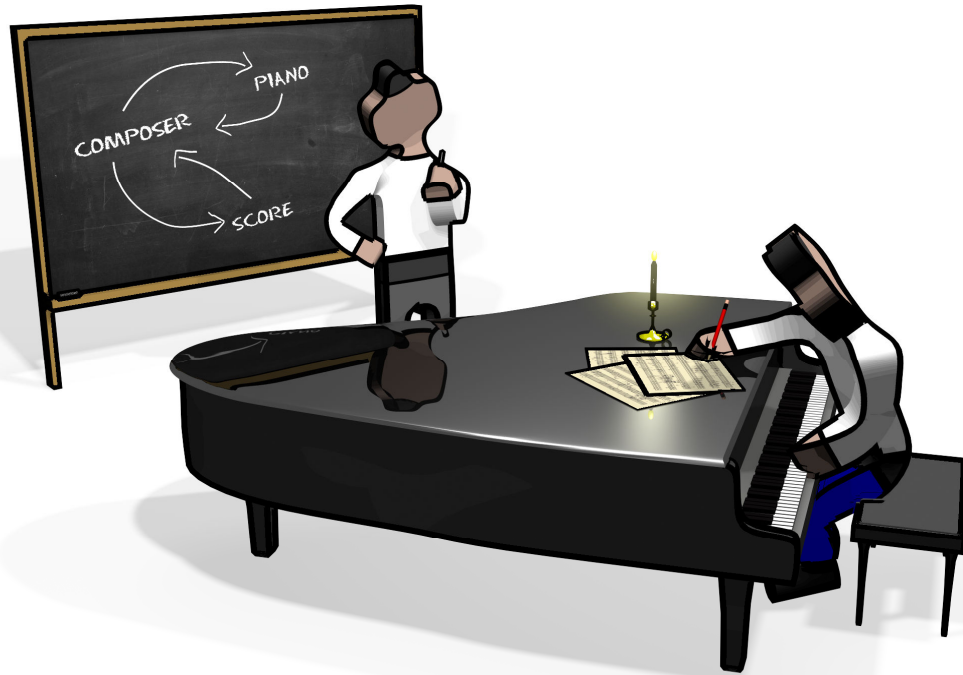
Table 4 – common components of flow (based on Csikszentmihalyi, 1996)

	importance of maintaining interaction “speed” and minimising interruptions, and specifically highlights how this can be facilitated by the learning and skill supported by the computer keyboard, as further explored in Chapter 7. ⁶⁵
<i>flow in music</i>	Flow is an integral part of musical experiences (Byrne <i>et al</i> , 2003; Chaffin and Limieux, 2004; MacDonald <i>et al</i> , 2006; Fritz and Avsec, 2007; Mullett, 2010) – “all musicians experience this creative peak in one way or another.” (Boyd, 1992; see footnote 63). Leman (2008) looks towards music technologies “as an

⁶⁵ Bederson (2004) describes the flow components supported by *NoteLens*, a simple note-taking program to support the quick recording and recollection of ideas, which uses keyboard shortcuts to provide a scalable user experience, enable fast expert interaction, and maintain focus on the task by avoiding visual distraction. This can be seen to mirror the interaction style of soundtrackers (see 2.2).

extension of the human body to reach peak experiences”, drawing on *embodied interaction* (Dourish, 2001), through the linking of motor skills and immersion in sound, to enable the optimal experience of flow. Specifically, he warns that if actions become decoupled from their response, “skills and challenges may become decoupled and interest may be quickly lost if the subject has the impression that improvement of skills has no apparent effect on feedback from the interactive system.” (Leman, 2008)

In the next chapter, flow theory and its components, along with the other themes reviewed in this chapter, are used to develop a model of the creative user experience in music composition. Appendix B also provides an overview of the components of flow in the context of musical creativity, drawing on writings on the experiences of composers and song writers (Graf, 1947; Sloboda, 1985; Boyd, 1992; Harvey, 1999).



chapter four **towards digitally-mediated creativity**

The previous chapter presented an overview of research into creativity, expertise and motivation, also identifying several limitations in the creative user experience afforded by modern music software. These user experiences are hard to describe and analyse using conventional, goal-oriented HCI design and evaluation techniques, where performance equates to speed and precise formulations of tasks are often required, yet are elusive in creative activities (Stowell *et al*, 2009). Following recent moves toward *user experience design* (e.g. Norman, 1993), several researchers have suggested the use of “flow” theory, in designing tools that support creativity (Bederson, 2004; Shneiderman *et al*. 2005; see section 3.7) – for which more “holistic” HCI frameworks, such as the *Cognitive Dimensions of Notations* (see Green and Petre, 1996), might provide a foundation.

In this chapter, I develop this approach and present a model of computer music interaction, designed to assist the analysis and design of creative user experiences, drawing chiefly on the concepts of *virtuosity* and *flow*. Common computer music scenarios are discussed in the context of the model, which is used to better understand the issues facing the interface designers of associated programs. In the next chapter, this approach is used to guide a large-scale, real-world user study.

The content of this chapter is to appear in the forthcoming *Oxford Handbook of Interactive Audio* (ed. Collins, K). See Nash and Blackwell (2012) in bibliography.

4.1 supporting virtuosity in computer music

The *Oxford English Dictionary* describes a virtuoso as “one who has special knowledge or skill in music”. In this research, virtuosity is defined as the enabling factor of fluency in a domain (i.e. music composition) through mastery of a device or system. As detailed earlier, such fluency is seen by researchers as one of the enabling factors of creativity (Amabile, 1993; Plucker and Renzulli, 1999; see Section 3.4).

*device-specific
knowledge*

Distinct from other types of knowledge in musical creativity (such as music literacy), this definition of virtuosity focuses on the development of interaction skill. Such *device-specific* knowledge concerns the learnability of a notation or input device, and notably the low-level aspects of interaction, supported by procedural memory and motor learning (see section 3.6). In this approach, expertise is developed with the notation, rather than the domain. With experience, actions become increasingly reflexive and automatic, allowing greater attention and focus on the domain itself (i.e. the music).

transparency

In this way, virtuosity is an approach to interface *transparency*, where “the user is able to apply intellect directly to the task” (Rutkowski, 1982; Holtzblatt, Jones, and Good, 1988). *Direct manipulation* is another approach to transparency (Shneiderman, 2005), extending the idea of visual metaphor to not only represent objects in the domain, but also how they interact with each other, creating a virtual, metaphorical world (Dourish, 2004). Recently, the use (and misuse) of metaphor has come to dominate usability design practice (Blackwell, 2006) – as can be seen in sequencers (see Section 2.1). Virtuosity, by contrast, is aligned with embodied approaches to transparency, such as those of Winograd and Flores (1987) and Dourish (2004), and relies as much on the physical interaction, as the visual aspects of the notation.

Following a similar approach, in the context of music; Leman (2008) warns that notation-mediated interaction models lead to a situation where the notation (and thus its designer) dictates the user’s perception of the domain. Sloboda (1985) also observes the historical influence of notation (both positive and negative) on music over the centuries. Accordingly, Leman advocates “direct involvement” in music, by minimising or removing notation from interaction; engaging with the physical domain itself, rather than a virtual or metaphorical representation. However, while apt for musical performance, it is less practical to cut notation out of the composition process, where its abstraction power provides the composer with broader editing scope and control of musical time.

Instead, virtuosity in composition is about learning how *changes* in the notation affect the domain, rather than simply how a static abstract representation maps to an end product. Thus, while a composer interacts with the notation, concrete feedback from the domain should be prioritised over more abstract visual feedback from the notation. In music, this means making the results of an edit available as sound, in addition to a visual, syntactic change in the displayed music notation (e.g. score). Skill development with the notation might then be driven by a phenomenological *hear-understand* process, rather than more structured formalisms or learning of theory.

4.1.1 design heuristics for virtuosity-enabled systems

Following the principles above, this section suggests design heuristics for developing interfaces to support virtuosity, drawing on literature reviewed in Chapter 3 and explored in subsequent chapters, through studies of skill learning (Chapter 7), feedback liveness (Chapter 8) and flow (Chapter 9) in music software.

*heuristic
evaluation*

Designing for virtuosity represents a different challenge to that of usability, and thus the heuristics presented here differ from, or even contradict, Nielsen's original recommendation for heuristic evaluation of usability (Nielsen and Molich, 1990; Nielsen, 1993), and other usability design manuals (Shneiderman and Plaisant, 2005; Sharp *et al*, 2007). Designing *multi-layered interfaces* to suit both novice and expert presents design challenges (Kitzmann, 2003), but a distinction is made in the targeting of expert users; a virtuosity-enabled system should enable a novice user to become expert – it does not rely on domain expertise learnt elsewhere (e.g. music literacy), though should consider the transferability of skills learnt.

*creativity support
tools (CST) workshop*

At points, these heuristics draw and develop upon the recommendations of a recent workshop report on *creativity support tools* (Resnick *et al*, 2005). These are discussed and referenced as appropriate, at the points denoted by the *cst* marker.

*cognitive dimensions
of notations (CDs)
correlates*

Similarly, the *cds* marker denotes a discussion of the each heuristic with respect to the *Cognitive Dimensions of Notations (CDs)* framework (Green and Petre, 1996). In Chapter 9, this framework is also used to establish *usability profiles* for creative programs, and explore the role of notation in supporting *flow*.

*the relationship
between heuristics*

Lastly, though each heuristic stands alone, attention should be drawn to the relationships between them. An effort has been made to highlight these interactions; in both the heuristics' descriptions and the order they are presented. As such, while a given heuristic

might propose design goals and objectives for the interface designer, subsequent heuristics can offer strategies to address these new challenges; engendering a gradual progression from design principles towards more specific design implications and manoeuvres. For example, skill development (H₁) can be facilitated by fast feedback loops (H₂), both of which can, in turn, be aided by simple interface primitives (H₃).

*design heuristics for
supporting virtuosity
in computer use*

*H₁: Support learning, memorisation, and prediction
(or “recall rather than recognition”)*

Expert interaction is enabled by the use of memory (Chase and Ericsson, 1981; Ericsson and Kintsch, 1995). While some interface widgets support both novice and expert interaction (e.g. the use of mnemonics, in menu accelerators), provisions for usability (e.g. “recognition rather than recall” – Nielsen and Molich, 1990) can hamper experts (van Dam, 1997) and their impact should be considered carefully in systems designed for experts. Using memory, interaction is no longer mediated through visual metaphors fixed by the interface designer, but by schemata derived from physical interaction and personal experience (Chase and Ericsson, 1981).

Notations should not aim or hope to be “intuitive”, or rely heavily on *domain-specific knowledge* (e.g. music literacy), or otherwise devalue the learning experience. Instead, they should provide a rewarding challenge that scales with user experience (Csikszentmihalyi, 2000). Corresponding studies of motor skill and learning in keyboard use in tracker interaction are provided in Chapter 7.

cst: The CST workshop likewise highlights the importance of expertise, in the development of tools for creativity. Such tools should present a “*low threshold*” (be accessible to the novice), whilst supporting a “*high ceiling*” (allow advanced uses). What distinguishes their recommendations from Shneiderman’s own calls for *multi-layer interfaces* (2005) is the addition of a third goal – “*wide walls*”, the support for a wide range of explorations, paths and interaction styles.

The design implications of this are the use of “very general primitives” (Resnick *et al*, 2005) in the interface (see H₂), which are themselves easily learnt, but can be efficiently combined or layered to create more complex functionality. Though this addresses design pitfalls inherent in *multi-layer interfaces* (such as over-simplification vs. over-specification, in respectively

catering for novices and experts), the workshop report cautions that a learning onus remains, as the users learn how to combine primitives. However, in virtuosity (and flow – Sections 3.6 and 3.7), learning is a desirable attribute in the creative user experience; and, in contrast to the steep learning curve of many professional music tools (based on knowledge of studio and music practice, see 2.1), a “simple primitives” approach provides a gradient that scales with experience.

CDs: CDs offer little account of ‘learnability’ as a factor of notation use (Elliot, Jones, and Barker, 2002), despite its relative importance in interaction design (Dix *et al*, 1998). Though the activity of *exploratory understanding* is identified as a possible goal of a notation, it concerns the learning of a domain. The literature makes few references to the learnability of notations, observing that greater *consistency* facilitates learning, *closeness of mapping* aids knowledge transfer, and *hidden dependencies* demand *long-term working memory* skills (Green and Petre, 1996). In our case, while the *visibility* of data in the notation remains crucial, parts of the interface might be hidden from view, if they can be triggered from memory (e.g. using shortcut keys). Likewise, the learning of more concise syntax provides for reduced *diffuseness*.

It can be argued that the challenge posed by learning constitutes a *hard mental operation*, as the user is encouraged to “work out more in their head” (Blackwell and Green, 2000), rather than relying on automation or notational hints, such as visual cues. An interaction requiring more mental effort and reflection can be more engaging and memorable, compared to calmer, more pedestrian interactions (Rogers, 2006). CDs literature to date, however, has focused almost exclusively on the negative design implications of *hard mental operations* (e.g. Green and Petre, 1996) – in contrast to most other dimensions, whose relative merit (or “polarity” – Blackwell *et al*, 2001) rests on context, or trade-offs with related dimensions. In the context of virtuosity, have we found a use case where *hard mental operations*¹, if not actually desirable, are tolerable?

¹ Perhaps more neutrally termed, *complex mental operations*.

H₂: Support rapid feedback cycles and responsiveness

To master a system, its behaviour must be “transparent” (Holtzblatt, 1988), allowing the user to easily equate cause with effect, in their interactions. In typical computer scenarios, basic control feedback should be provided within ~100ms (Miller, 1968; Nielsen, 1993) to appear instantaneous. Complicated operations should complete within roughly 1s (~300ms to 3s), or otherwise risk interrupting the flow of thought (Newell, 1990). After 10s of idleness, users actively become restless, and will look to fill the time with other tasks (Nielsen, 1993). As such, longer delays, especially those requiring wait cursors or progress meters, should be avoided; and are “only acceptable during natural breaks in the user's work” (Nielsen, 1994).

To support live performance and recording, there are even stricter criteria for music systems, which must respond within a few milliseconds (Walker, 1999).² Dedicated low-latency sound drivers, such as Steinberg ASIO and Microsoft WDM, were developed to provide such latencies – typically confining delays to under 25ms, and potentially as low as 2ms. Even below this threshold, musicians and professional recording engineers are sensitive to *jitter* (the moment-to-moment fluctuations of clock pulses, measured in nanoseconds), but the impact is perceived in terms of sound quality (the addition of noise and inharmonic distortions, and deterioration of the stereo image), rather than system responsiveness.

	timing	perceived as...	if violated...
Table 1 – timing requirements in computer music interaction	< 1 ms	sound quality ('tightness', 'jitter')	<i>user hears noise artefacts, inharmonic distortions, muddled stereo image</i>
	< 25 ms	realtime audio ('low latency')	<i>user has difficulty keeping musical time, maintaining sync. during performance</i>
	< 100 ms	'instantaneous' UI response	<i>system feels slow and unwieldy, harming user's sense of control</i>
	< 1s	noticeable delay	<i>user has difficulty planning ahead and maintaining “flow of thought” or continuity of action</i>
	< 10s	tolerable delay	<i>user loses focus, and their attention wanders to other tasks</i>

² Indeed, a realtime music system operates at much higher timing resolutions, up to 192kHz (1 sample every 5ns). Human hearing extends to around 20kHz, but these extensions account for the requirements and limitations of digital audio, such the *Nyquist limit* (dictating the 44.1kHz specification of CD Audio), *aliasing* (prompting the 192kHz specification of DVD Audio) and *oversampling* (to improve signal-to-noise ratio, particularly in cheaper hardware).

While less “live” interactions, such as playback control and general UI responses, tolerate higher latencies, longer delays nonetheless affect the perceived directness of the user experience. Table 1 summarises these requirements for interaction, in a musical system, with examples. The relationship of timing and control emerges; the finer the required control, the tighter the demands on responsiveness.

Another way of looking at this trend is to consider the relative availability and timeliness of feedback from the domain itself. In music, “live” interactions are not only highly responsive, but are also driven by *concrete feedback* from the domain itself – that is, raw feedback in a form not encoded in or constrained by the abstract formalism of a notation. As visual notations, UIs provide *abstract feedback*, presenting only a representation or specification of the end product.

A related concept of “liveness” exists in programming (Tanimoto, 1990; see Section 4.2.4).³ Like music composition, programming is the process of specifying and scripting future events (Church, Nash, and Blackwell, 2010). Programmers use various forms of abstraction to describe a program, which is then compiled or interpreted to executable code. Liveness is a quality of the design experience that indicates how easy it is for a programmer to get an impression of the end product, during various stages of design. More generally, promoting liveness is an example of the push to accelerate the feedback cycle in software design, complementing the philosophies of similar moves towards *rapid application development (RAD)*. In RAD, the early-availability of testable prototypes allows more flexible targets, and facilitates experimentation and ideation (Resnick *et al*, 2005), both of which constitute enabling factors of creativity (Sternberg, 1999; see 3.1).

In a computer music context, “liveness” thus means being able to easily audition the music encapsulated in the visual notation (e.g. specific notes, phrases, instruments) (Nash and Blackwell, 2012). Music notations tend towards an “eager linearization” of time (Duignan *et al*, 2005), scripting a linear sequence of musical events. Excerpts of single beats, bars, phrases,

³ In music, the term “liveness” is increasingly used to describe a subjective sense of intimacy and immediacy in live art, as experienced between audience and performer (e.g. Auslander, 1999). In live electronic music, research highlights the challenge of delivering liveness in the context of disembodied, acousmatic sound (e.g. from a laptop), decoupled from a performer’s physical actions (Emmerson, 2007). Though this use of the term differs from that used in this thesis (i.e. Tanimoto, 1990; discussed in Section 4.2.4), the two contexts are related: When liveness is lacking, the audience/user feels less a part of the performance/music, and may find it harder to understand what they hear or should expect, given the (limited) visual feedback.

movements, and even arbitrary segments of the music, are easily evaluated. As a result, realtime music systems can harness the principles and benefits of fast feedback loops at various granularities, to foster iteration and experimentation. The use and role of feedback (and liveness) in modern music software is examined in detail, in Chapter 8.

cst: Providing support for exploration was the primary design principle put forward by the CST workshop report, which advocated the use of ‘what-if’ scenarios and an emphasis on iterative design, experimentation and “tinkerability” in the user experience – allowing users “to mess with the materials, to try out multiple alternatives, to shift directions in the middle of the process, to take things apart and create new versions.” (Resnick *et al*, 2005) Papert’s related concept of “bricolage” observes how such a *constructionist*, experiential approach to learning provides a more personal, flexible alternative to the traditional “analytic, rule- and plan-oriented style” (Turkle and Papert, 1992).

cds: In the CDs framework, *exploratory design* is one of 6 basic activity types, each of which entail distinct dimensional profiles (Blackwell and Green, 2003). The *progressive evaluation* dimension measures how evaluable the end-product is, at various stages of development. Providing for it can mitigate the *premature commitment* of requiring a completed work before feedback is available.

The opportunity to then take a new direction falls under the dimension of *viscosity* (resistance to change). A creative environment must be non-viscous, supporting all manner of changes at any stage in design (Gentner and Nielsen, 1996).⁴ Green and Petre (1996) make the distinction between *knock-on viscosity*, where small changes require the user to repair consistency of their work, and *repetition viscosity*, where multiple (repetitive) actions are required towards a single goal. In each case, the UI designer has the opportunity to automate or abstract such involved or laborious processes. Again, there is not only the risk of concealing the inner workings of the system, making them hard to grasp (see H₃), but also of reducing the user activity and engagement surrounding the task, as well as their perception of being in control.

⁴ *Modelessness* (Gentner and Nielsen, 1996; see H₄) can also be seen as facilitating faster changes to notation (e.g. without the *premature commitment* of changing mode).

H₃: Minimise musical (domain) abstractions and metaphors

In HCI, UI designers try to reify the user's "mental model" to represent and operationalise a task domain (Norman, 1988), using predetermined abstractions (Duignan *et al*, 2005) and metaphor (Blackwell, 2006), across various levels – processes, properties, states, relationships. The formalisms of any notation determine the expressive flexibility it allows, shaping a user's perspective of the domain, or even an entire culture's (Sloboda, 1985). It is difficult for a UI designer to match the user's internal representation of musical expression without inadvertently shaping or constraining their creativity (Cascone, 2000; Kitzzmann, 2003; Duignan *et al*, 2005).

Though classically-trained or musically-literate users will share many perceptions of musical structure, there are few widely-accepted formalisms encapsulating the full gamut of computer music capabilities, and non-digital abstractions (such as wires, pots, pedals, mixers, or other metaphors to electronic and acoustic music; see Section 2.1) can be confusing, confining or cumbersome (Desain *et al*, 1995; Duignan, 2007).

Indeed, it is a major challenge for UI designers to design any *unified* user interface for artistic audiences, who define themselves by their uniqueness and innovation – how do you design a box, for people who want to think outside of it?

One approach is to simply make the box smaller: avoiding the use of higher-level abstractions, in favour of low-level primitives, that can be layered and combined, by the user, to produce equivalent or greater functionality. The simpler functionality of each primitive makes it easier to understand and learn. Then, as more are layered and combined, the challenge increases, providing a scaleable learning experience, towards the development of broader mastery. Turkle and Papert (1992) call this "soft-mastery", observing that it encourages "closeness to the object", and that such bottom-up perspectives are common in fine artists and musicians.

Automation, as an abstraction of process, should also be considered in respect of keeping the user active and engaged. An interaction designer, in automating trivial yet laborious tasks can increase overall productivity and reduce the effort invested by the user. However, this also has the effect of reducing their involvement in the workings of the system; harming their understanding of the system (*sense of control*), or leading to periods of waiting and idleness (see H₁), interrupting the continuity and flow of interaction.

cst: The CST workshop report similarly concluded that achieving the desired *low threshold*, *high ceiling*, and *wide walls*, in creative support tools, revolved around careful selection of “black boxes” and keeping interface primitives as simple as possible (Resnick et al., 2005).

Resnick *et al* (2005) also warn against “creeping featurism”; the tendency to incrementally add advanced features in new software versions, encouraged by the ease of selling products based on feature set, rather than user experience. This trend is evident in production tools like DAWs (Duignan, 2007; see 2.1), which introduce specialist tools or processes, sometimes as black boxes, raising complexity and reducing consistency.

cds: Many modern music programs can be described as *abstraction hating*, favouring predefined, standardised objects that facilitate *out-of-the-box* use, assuming the user is already familiar with the conventions used – metaphors from the analog recording studio (mixers, wires, faders, etc.) or acoustic music (pianos, the score, etc.). In such cases, “the *closeness of mapping* to conventional audio processing equipment ... is indicative of a corresponding reduction in potential for creative exploration” (Blackwell and Collins, 2005). Duignan (2007) also details several issues with the abstractions presented in existing music production software.

Notations that encourage users to form their own abstractions are described as *abstraction tolerant*. In the literature, this normally implies the use of a *redefinition device*, allowing users to explicitly change the notation to suit their interaction style or perception of the task domain (e.g. Blackwell & Green, 2000). The process requires that the user is not only familiar with the definitions of the original notation, but can also articulate the appropriate *re-definition*, using a separate notation. As such, it can require an enormous *attention investment* (Blackwell, 2002) and encourages precisely the planned, analytic interaction style we wish to avoid – for periods, interrupting the user’s workflow, increasing their exposure to predefined formalisms, potentially distancing them from the music itself.

Instead, while still favouring notations that are *abstraction tolerant*, this heuristic proposes less formal abstraction processes. To this end, *secondary notation* (some means of including extra information, other than formal syntax – e.g. annotations, comments) may offer suitable interpretive freedom. However, the interface designer should also not

overlook the implicit abstractions a user will make by simply listening to their music – the user’s perceived structure of the sound itself. In this capacity, simply the broad availability of such musical feedback (see H₂), may be the best scaffolding for a user to form their own abstractions. In this sense, such ‘audibility’ may be seen as a correlate of the *visibility* dimension (concerning how much systems “bury information in encapsulations”; Blackwell and Green, 2003) and, accordingly, the *closeness of mapping* dimension (“How closely does the notation correspond to the problem world?” - Green and Petre, 1996) thus concerns only the selection of low-level primitives.

H₄: Support consistent output and focused, modeless input

An interface that remains consistent, from moment to moment, can be more easily remembered and predicted. Fixed, static layouts enable the development of not only spatial schemata, but also motor learning (Smyth *et al*, 1994), both of which allow a degree of interaction to be handled subconsciously (see H₁).

Changeable, dynamic screen layouts (such as floating windows) require conscious reflection, interrupting thought processes and hampering the performance of experienced users. Whereas inexperienced users want to “find where everything is” (Kitzmann, 2003), experienced users want to *know* where everything is. They should not have to visually search through different windows, modes or other views, to locate information or effect minor edits, but “should be able to perform any task at any time” (van Dam, 1997).

Novice users, on the other hand, may benefit from taking their initial steps more slowly, and digesting the program in smaller chunks, or else find themselves daunted by the program’s surface complexity (Norman, 1988). Limited screen real-estate, even on high-resolution computer displays, forces interface designers to divide functionality across separate views – often exposing only part of the notation in each. Interface hierarchies, like menus and window systems, are used to breakdown complex programs into simpler parts, while presenting a logical ordering, that attempts to balance how easily a novice can identify the appropriate selection, against how quickly an expert can make it.

Most programs still present a primary notation that is kept in view for the majority of the time, and which constitutes the focus of activity – for example, the source code in an IDE, the

document in a word processor, the waveform in a sound editor. It is the first view that greets the newcomer, and the first they learn, before moving onto other parts of the program.

However, an equivalent notational focus is largely absent from many music production programs – perhaps because a definitive visual representation of digital music is so elusive (see H₃). In a DAW, workflow is spread over multiple windows and input devices (see Figure 2-1), serving various purposes. The *desktop studio* is exactly that; a studio on the desktop – a combination of separate interfaces mimicking the separate “devices”⁵ in a recording studio, through visual metaphor (Duignan *et al*, 2005), wired together to allow you to capture and mix a musical performance. To become expert in these programs, you must become expert in the full range of hardware and processes used in the electronic studio – tasks usually shared across several individuals, including performers, tape operators, sound engineers – each able to focus on a specific device. There is little consistency across the different devices, at the same time providing diverse editing techniques to achieve similar end results (see Table 2), while occasionally offering select capabilities found nowhere else in the environment.

In addition to spreading functionality over different areas of the program, a single-user studio paradigm encourages the segregation of the music-making process over time – prepare, perform, record, edit, mix, finalise. Each stage depends on the previous, and requires the user to have a clear, preformed concept of what they want to achieve. Exploring and experimenting with different ideas involves moving back-and-forth between these stages and interfaces.

Table 2 – Controlling note volume in DAWs,
a list of selected settings and associated interfaces that influence the final volume of a single note, in a standard DAW setup

variable or setting	controlled using
MIDI note velocity	<i>MIDI controller, piano roll, score, data list</i>
MIDI key/channel aftertouch	<i>MIDI controller, arrange window, data list</i>
MIDI channel volume	<i>arrange window, mixer, data list</i>
MIDI track volume	<i>arrange window, mixer</i>
MIDI excerpt volume	<i>arrange window</i>
volume envelope settings	<i>synthesizer</i>
MIDI global volume	<i>synthesizer</i>
master volume setting	<i>synthesizer</i>
audio input gain	<i>soundcard setting</i>
audio track gain	<i>arrange window, mixer</i>
audio track level	<i>arrange window, mixer</i>
audio output bus level	<i>arrange window, mixer</i>
master volume	<i>mixer, transport bar</i>
audio output level	<i>soundcard setting</i>

⁵ The actual term used by *Steinberg Cubase*, for each editing view.

Hardware devices (e.g. MIDI instruments, controllers, control surfaces, digital mixers) can make visual metaphors tangible and enable peripheral interaction (Edge, 2008), often presenting fixed, physical layouts that aid motor learning. At the same time, they potentially swap the contention of screen space for that of desk space, risk confining the user's control and attention – their hands and eyes – to specific devices, and increase the effort of moving back-and-forth during experimentation. Such tradeoffs are evident in computer music hardware and studio equipment (see Section 2.1). The use and role of visual focus and feedback in music software is explored in Chapter 8 (Sections 8.4 to 8.5).

This heuristic calls for a more central focus in computer music interfaces, and principled separation of primary and peripheral notations. This model already exists in some computer music practices, such as *live coding* (Blackwell and Collins, 2005) and *score editing*. In score editors, the notation (the musical score) is not always apt for representing computer music and digital audio processes.

CDs: This heuristic is closely related to the cognitive dimension of *consistency*, which has been linked to the learnability of a notation already (see H₁; Green and Petre, 1996). The DAW's variety of notations can be seen to improve *role expressiveness* (Blackwell and Collins, 2005), albeit at the expense of such *consistency* – potentially demonstrating a trade-off between these dimensions.

Minimising *diffuseness* improves *visibility*, reducing the need for keyhole editing techniques, such as multiple windows and scrollable views, that selectively show or hide parts of the notation. Such divisions of information, across different modes or views, can likewise create *hidden dependencies* (as in Table 2), obfuscating the behaviour of the system.

Finally, the need to complete tasks in a specific order represents a *premature commitment*, and increases *viscosity*, which is undesirable in *exploratory design* activities, such as musical creativity (Blackwell *et al*, 2000).

These heuristics detail specific goals and properties that creative systems should have to support the development of virtuosity. In the next section, we expand upon the relationships between them, integrating them with a broader concept of flow (Csikszentmihalyi, 1990), to develop a more general model of the creative user experience.

A considerable amount of existing HCI design and evaluation methodology relies on the articulation and abstraction of objectives, success criteria, tasks and processes. Ill-defined creative pursuits, like music, rarely permit this; a piece of music need have no practical purpose and is only deemed satisfactory or complete if the composer says so (see Section 3.2). They might start writing at any point in the piece, for any instrument, at any pitch, any volume, etc. and they might then return to that point at any time subsequent, and change it to something completely different. Of the computer, they demand the creative freedom to be able to do anything to anything, at any time, in any order. Each action informs the next, with the artist perpetually seeking to maintain their creative thread, flow, and pace.

This section builds on the previous, extending concepts and proposals concerning ‘virtuosity’ to develop a generic model of the creative process in music that can act as a framework for discussing and designing open-ended user experiences that support flow (Section 3.7) and creativity in general (Chapter 3).

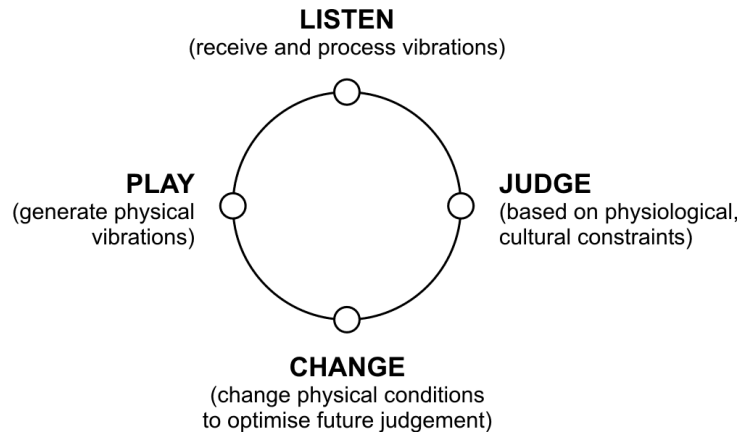
4.2.1 from virtuosity to flow

The description of the creative user experience presented above shares characteristics Rittel and Webber (1973) outlined for “wicked problems” (Section 3.2), except that creative endeavours, in contrast to being “one-shot operations”, allow for and demand more experimental trial-and-error learning and practice. Outside of a live, public performance, musicians and composers can try new ideas without significant consequences, or the worry of failure.

feedback cycles

In Section 4.1, support for rapid feedback cycles (H_2) was proposed as a way of supporting such experiential learning, and the development of computer music *virtuosity*. Leman (2008) expands on the idea of the feedback loop in music, identifying four repeating stages: *play*, *listen*, *judge*, and *change* (Figure 1). He uses this ‘action-reaction cycle’ as the basis for a philosophical framework for computer-mediated *embodied music cognition*, which he proceeds to use as an argument for the role of gesture in music interfaces. Leman argues that the approach has the power to afford “direct involvement” in music, by cutting out the indirection inherent in conventional interfaces. In many ways, his thesis can be seen as a musical reworking of *embodied interaction* (Dourish, 2004), drawing on the phenomenological approaches, pioneered by Heidegger and Husserl, and introduced to the wider HCI community by Winograd and Flores (1986).

Figure 1
Action-reaction cycle
(Leman, 2008)



*'readiness-to-hand',
transparency, and
the "flow of work"*

Karen Holtzblatt's earlier work (on what later developed into *contextual design*) similarly drew on this foundation, to discuss Heidegger's 'readiness-to-hand' as an approach to interface *transparency* (see Section 4.1) – advocating "creative iteration", the maintaining of "workflow", and avoidance of disruptions, in the user experience (Holtzblatt *et al*, 1988) – foreshadowing subsequent HCI rationales for Csikszentmihalyi's own theory of flow (Norman, 1993; Bederson, 2004; Shneiderman *et al*, 2005).

*virtuosity as a
basis for "flow"*

Flow, the theory of "optimal experience" (Csikszentmihalyi, 1988; 2000 – detailed in Section 3.7), describes a mental state that underlies creativity (Csikszentmihalyi, 1996) and enumerates several components (see Table 2-4) required to achieve it, for a given activity. In addition to advocating rapid feedback (H_2), other aspects of the definition of *virtuosity* in section 4.1 are implicit in some of the specific requirements of flow (see Table 3) – both call for interactions supporting focus, concentration, skill development and directness of control and feedback.

*action-awareness
merging*

Of the two remaining components of flow (absent in Table 3), *action-awareness merging* objectifies the resulting trance-like flow state itself, where the individual is wholly-engaged in the activity, unfazed by external, environmental factors. It can thus be seen as the product of other components, such as *loss of self consciousness* and *concentration and focus*, as well as a corollary of *distorted sense of time*.

*intrinsic motivation
and rewards*

The other missing component similarly stresses a separation from the outside world – that the activity be *intrinsically-rewarding*. Flow theory was developed upon the idea of *intrinsic motivation*, which occurs in an activity that is its own reward, in contrast to tasks requiring external incentives (*extrinsic motivation*; e.g. deadlines, penalties, money, recognition – see Section 3.7). An *intrinsically-rewarding activity* is thus an enjoyable task that is both fun and fulfilling.

Table 3 – the design heuristics for virtuosity, and corresponding components of flow

<p><i>H₁: Support learning, memorisation, and prediction (or “recall rather than recognition”)</i></p> <p>→ <i>Balance of Challenge and Ability</i> A learnable notation allows interaction skill to develop, and for achievement to scale with experience (see H₃). By contrast, traditional usability heuristics, such as Nielsen and Molich’s (1990), advocate “recognition rather than recall” (<i>Minimize User Memory Load</i>) and minimal learning curves, tailored for novice use.</p> <p>→ <i>Clear Goals</i> Although final goals are hard to articulate, in creative endeavours, this component of flow concerns the moment-to-moment goals in interaction, and how easy it is for the user to discern the appropriate actions to take, to achieve a desired outcome. As such, it represents the latter stages of virtuosity, where mastery allows <i>transparent</i> use of the interface, balancing high levels of ability and challenge.</p> <p>→ <i>Sense of Personal Control</i> A notation that allows skill to develop empowers the user. Experience allows the user to predict what to expect so that they can plan ahead and actively drive interaction, rather than passively rely on visual cues and hints, allowing the program to set the pace.</p> <p>→ <i>Loss of Self Consciousness</i> When an interaction is learnt or memorised to the extent it becomes automatic, it becomes <i>reflexive</i> rather than <i>reflective</i>, and the user becomes less consciously aware of how such actions and behaviour appear to others, allowing them to focus on the task itself.</p>
<p><i>H₂: Support rapid feedback cycles and responsiveness</i></p> <p>→ <i>Direct and Immediate Feedback</i> Feedback should not only be fast, but direct from the domain. In a musical application, any change caused by an edit should be immediately reflected in the visual notation, and immediately available to audition, aurally.</p> <p>→ <i>Distorted Sense of Time</i> Slow or delayed feedback, such as those prompting idleness, wait prompts, progress bars or predicted completion times, implicitly or explicitly draw a user’s attention towards the passage of time. The user must be able to proceed at their own pace, <i>in their own time</i>.</p>
<p><i>H₃: Minimise musical (domain-) abstractions and metaphors</i></p> <p>→ <i>Balance of Challenge and Ability</i> The use of simple interface primitives, which are easily learnt individually, provides novice users with a low starting threshold, while enabling increasingly greater challenges and functionality, when such primitives are combined (Resnick <i>et al</i>, 2005).</p>
<p><i>H₄: Support consistent output and focused, modeless input</i></p> <p>→ <i>Concentration and Focus</i> Spreading interaction across multiple windows, views and input devices splits the user’s attention. Where modes or windowing are unavoidable, the user should be able to focus and concentrate on that area of program, without having to refer elsewhere.</p>

Musical performance is a commonly cited example of such an intrinsically-rewarding activity (Leman, 2008), and can be used to illustrate many of the flow components (Csikszentmihalyi, 1998; 2000) and obstacles to be overcome. Acoustic instruments, for example, offer *direct and immediate feedback*, allowing experiential learning and, with experience, total, reflexive control (Williamon, 2004). Musicians can lose themselves in their instrument, for hours at a time (Boyd, 1992).

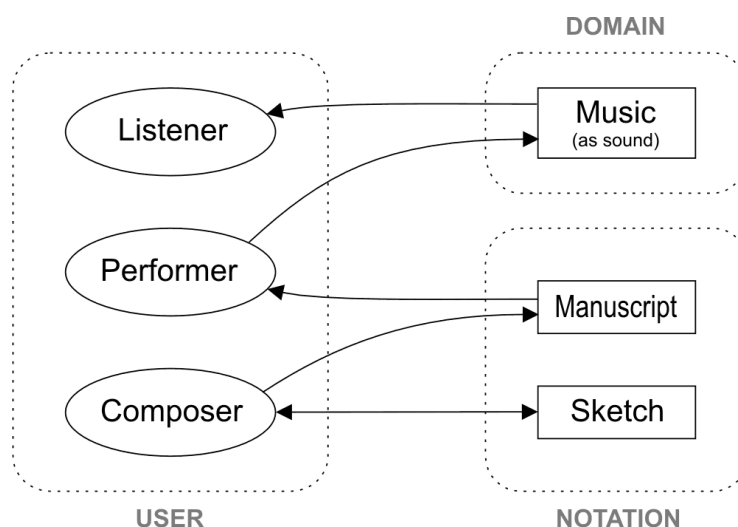
In the formative stages, however, a musician may have trouble articulating a pleasing sound, or achieving a *sense of control*. The lacking *balance of challenge and ability* can make them prohibitively *self-conscious*, especially if a suitably private practice space is not available. Similarly, music literacy, when taught as a pre-requisite to music interaction, presents an additional challenge for beginners that, as an example of “structured learning”, can detract from more playful and enjoyable “informal learning” experiences (Norman, 1993; see also Section 3.6), and deter students from persevering.

Ultimately, maintaining flow in the creative user experience requires protection from interruptions and distractions from the outside world, ensuring the user maintains focus, motivation and control.

4.2.2 abstracting the creative music process

Having observed the difficulty associated with a precise definition of goals and methods in a creative process, this section considers the various roles of different people and commodities historically involved in the composition process, in an attempt to articulate those of the computer-based composer.

Figure 2 - a contextual design “flow model” of work (music) in the composition process

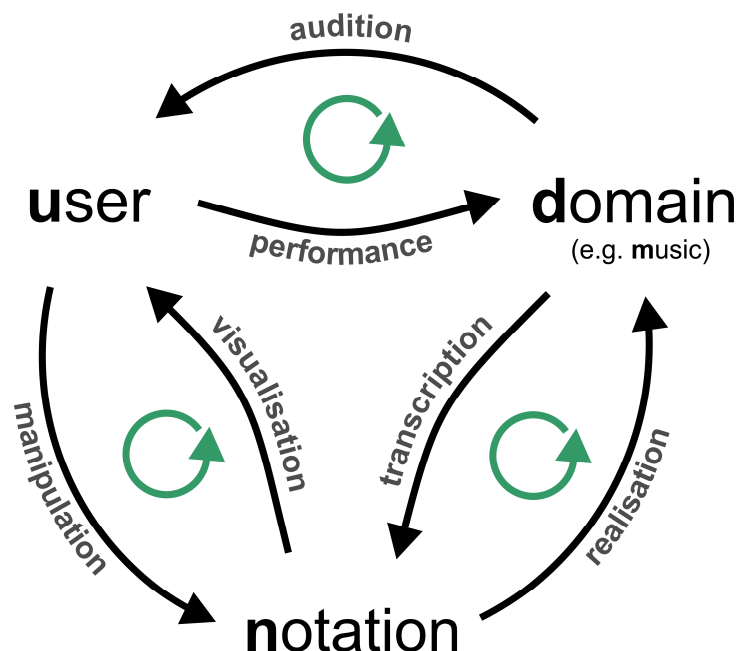


In *contextual design* (Holtzblatt, Wendell and Wood, 2004), this approach is known as *work modelling*, and is used to produce the “flow model” in Figure 2. Significantly, the modelled “flow” concerns the communication and coordination of *work*, and is not a reference to Csikszentmihalyi’s mental state.⁶ However, a shared foundation in phenomenology helps justify the model’s use here (see Holtzblatt, 1988). Shneiderman (whose recent work has focused on creativity – see 4.1.1) also comments that such modelling is a useful way of presenting a “structured process, with sufficient freedom for innovative excursions.” (Holtzblatt, Wendell and Wood, 2004). Here, the approach is used to give a basic structure to the largely unstructured process of innovation in music, further developed in the next section.

4.2.3 systems of musical flow

In Figure 3, I present a model to represent the creative process in music, in a way that can be operationalised for use as a tool in the design and evaluation of digital music interfaces, and that provides a basic taxonomy for talking about the creative process in interactive music systems.

Figure 3
the systems of
musical flow model
of digitally mediated
music interaction



⁶ According to Csikszentmihalyi (1988) the term “flow” originated from the analogy of being carried along by a current of water, as presented by several subjects in early interviews on the flow experience.

In the computer, the roles of composer, performer and listener are unified as that of the user, separately presented in the model of the previous section (Figure 2). Even in conventional acoustic practices, these roles are not mutually-exclusive – the composer (as performer) will audition musical phrases before committing them to notation, and (as listener) draws inspiration for new phrases by listening to existing material. At the same time, the computer also assumes the role of performer, able to realise the notation as sound programmatically. Accordingly, folding the original model to present the process from a single-user perspective creates the three-way dialogue between the user, notation and musical domain, seen in Figure 3.

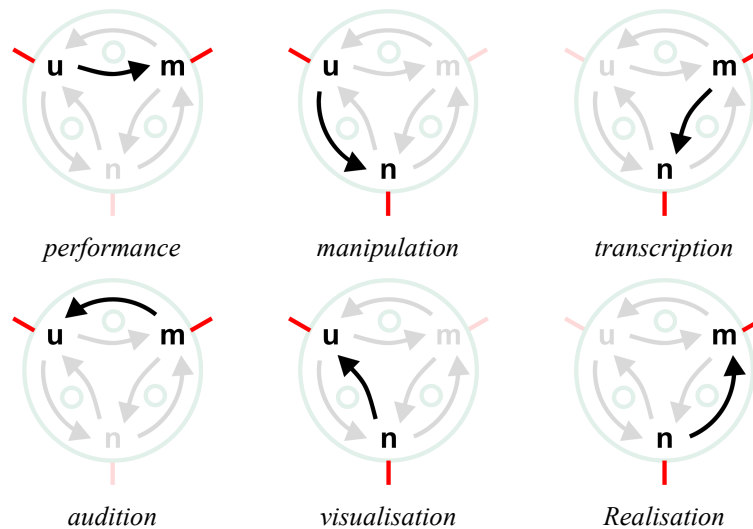
The approach is thus able to model the creative process as a network of feedback loops, such as those described earlier for *virtuosity* (see 3.1) and similar to the *action-reaction cycles* described by Leman (2008). Moreover, the *closed* nature of these feedback cycles, as well as the system at large, signify the *intrinsic* properties of interaction – where rewards and motivations arise from within the user experience itself, be it the music contained in the notation, or the sense of empowerment from mastering the system as a whole. By contrast, an interactive system typified by an *acyclic* graph (not illustrated in Figure 3, but discussed later in this section) signifies an *extrinsic* factor, which may indicate a problem with the interaction.

*conceptual
spaces*

The three nodes in the graph represent the *conceptual spaces* of the *user*, the *notation* and the *music*. In any space, a “concept”⁷ has the potential to motivate the creative process and enable flow. Each supports a distinct representation of the domain – perceptual, virtual, and real, respectively. Their mapping onto each other defines the creative user experience, as musical ideas are passed around the system and repeatedly translated (possibly not faithfully) from one space to another.

⁷ A precise definition of this term entails a discussion of one of the most debated topics in musicology; the signification (meaning) of music. As will become apparent, this model does not rely on a specific interpretation of this term, in order to be useful as an *engineering solution*; and only observes that such “concepts”, whatever they may be, are those that motivate and pre-empt action. For further reading, the topic is thoroughly explored in several texts: in computing (Winograd and Flores, 1987), in interaction (Dourish, 2004), and in music (Sloboda, 1985; Leman, 2007; Cross and Woodruff, 2008).

Figure 4
creative sub-processes



*creative
sub-processes*

The arcs (see Figure 4) represent *creative sub-processes* that translate concepts in one space to those of another and, in so doing, shift the motivational impetus. The *creative sub-processes* (*performance*,⁸ *audition*, *visualisation*, *manipulation*, *realisation* and *transcription*) form the basic building blocks of a creative music process. For example, a musical concept in the *user* is transformed by a process of data *manipulation* into musical data in the *notation*, whereupon it can be transformed either by *visualisation*, back to the *user*, or through a process of *realisation*, into some physical instance of *music* (e.g. sound).

*intrinsic
vs. extrinsic*

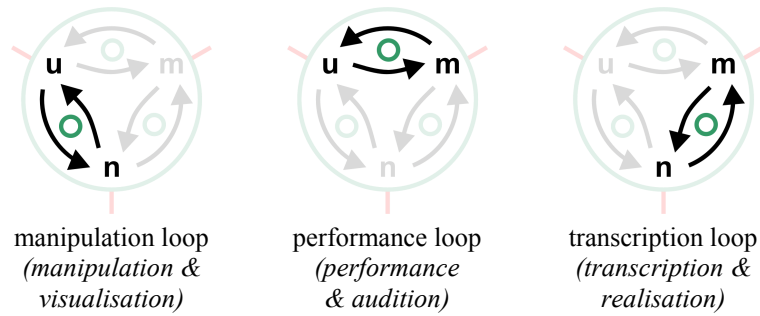
A *system* is formed by the combination of two or more creative sub-processes, modelled as a directed graph. By considering the cyclicity of the graph, designers and evaluators can predict whether, and to what extent, the system affords motivation *intrinsically*, and whether it depends on *extrinsic* factors (an external goal, reward, or other incentive). Intuitively, cyclic, closed loops help contain motivation within the system and preserve flow, whereas acyclic, open paths require external sources and sinks. Such paths introduce uncertainty, self-consciousness, as well as a perceived loss of control, and thus typically impede flow (Csikszentmihalyi, 1990).

a scalable model

Using these graphs, it is possible to model, diagnose and design user experiences of varying complexity, accounting for multiple notations (a skill-intensive example of *macro-flow*; Csikszentmihalyi, 1992) and multiple users or systems (potentially affording *group flow* – see Csikszentmihalyi, 1990). Most real-world scenarios, however, can be discussed in terms of simpler configurations.

⁸ As discussed in Section 3.1, the terms ‘performance’ and ‘audition’ are used in a technological context only, referring to any time-critical execution of a task or direct interaction with a domain (“Level 4 liveness”, see Section 4.2.4) and realtime evaluation of the product, irrespective of social context (e.g. the presence of listeners other than the practitioner, such as audience or collaborators).

Figure 5
intrinsic micro-
processes



*intrinsic micro-
processes (basic
feedback loops)*

*composite
processes*

Three basic feedback loops (see Figure 5) form *intrinsic micro-processes* between the *user* and *notation* (the *notation loop*), the *user* and *music* (the *performance loop*), and the *notation* and *music* (the *transcription loop*).

Arcs can also be paired to form *composite processes* that are, by themselves, extrinsic. As can be seen in the symmetry of Figure 6, composite processes mirror the function of corresponding creative sub-processes. For example, the *user* can affect *music* either directly through *performance* or, indirectly through the *notation*, using the transitive coupling of data *manipulation* and *realisation*. The difference between the two paths is the difference between Leman's aforementioned "direct" and "indirect involvement" in music (Leman, 2008). However, as is easily deduced from Figure 6, one or more composite processes are necessary if a system is to encompass all three conceptual spaces, which are necessary when the activity is, for example, composition rather than performance. Indeed, the introduction of notation, into the last example, would seem to transform a process of *performance* into one of *structured composition*. Figure 6 illustrates the six composite processes, each corresponding to one of the six creative sub-processes, in Figure 4.

Figure 6
composite processes
with constituent
(and equivalent)
sub-processes

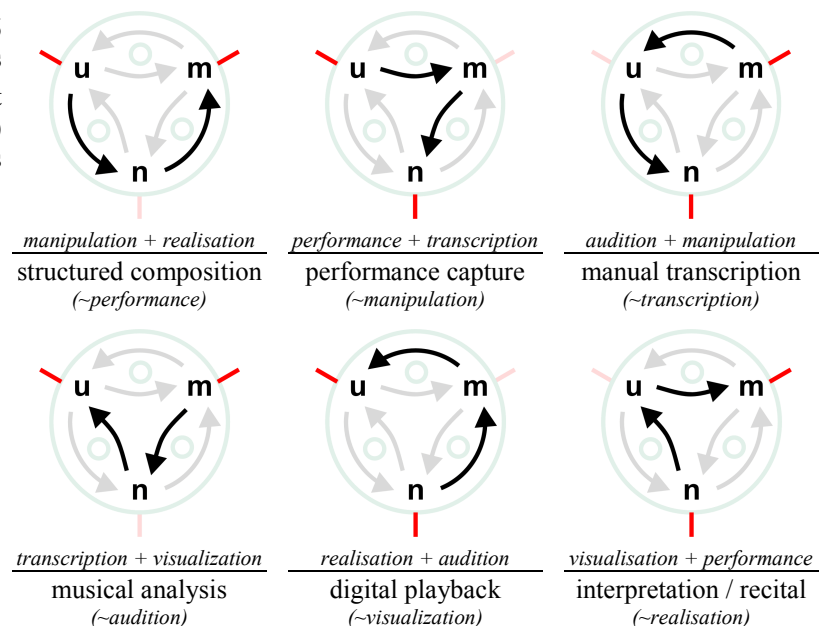
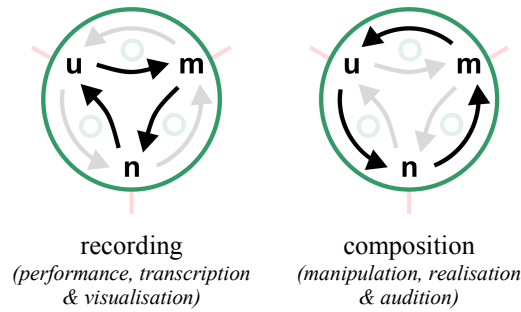


Figure 7 – intrinsic macro-processes



Paired, sub-processes afford either intrinsic couplings of two conceptual spaces or extrinsic couplings of all three. To incorporate the *user*, *music* and *notation* in an interactive system supporting flow, three or more sub-processes must be combined, to form a closed loop, intrinsic system.

*intrinsic macro-
processes*

Figure 7 illustrates how three *sub-processes* are combined to form one of two ***intrinsic macro-processes***, either of which can form the foundation of a flow-enabled interactive system. Clockwise, the loop entails *performance*, *transcription*, *visualisation*, describing a ***recording*** process. Anti-clockwise, the loop entails *manipulation*, *realisation*, *audition*, describing a ***composition*** process. Loops can also be seen as combinations of respective composite processes.

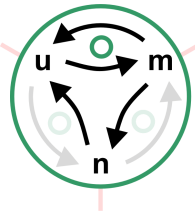
system archetypes

With four *sub-processes*, systems support two intrinsic loops, yielding nine ***system archetypes***. Most real-world music activities can be considered in terms of one of these archetypes, enabling flow in different ways, with various trade-offs, as detailed in Figure 8.

*operationalising
the model*

In prototyping an interactive system, a designer must decide which arcs and loops are desirable in their user experience, and how they can inform the design. Accordingly, an evaluator decides if specific arcs or loops are both present and adequately implemented in a system under evaluation. Some specific design trade-offs are identified in Figure 8, but a more general vocabulary is needed if the model is to be scalable to a broader and more complex variety of scenarios. The system features of the model (Figure 9) are the key to its operationalisation and verification as a design and evaluation methodology. Such system features can be considered in terms of specific conceptual spaces, processes or feedback loops, to obtain a more detailed system description. For example, a “multi-modal system” can be considered in terms of *flow redundancy* (multi-modal feedback), *flow fission* (multi-modal action) and *flow congestion* (multi-modal feedback and action). As such, the system has the potential to inform or confuse the user’s comprehension of the mapping between the *notation* and the *music*. Alternative, a “cross-modal system” (*visualisation* and *performance*, or *audition* and *manipulation*) must consider the same factors, but also *flow estrangement*, which can further obfuscate the mapping.

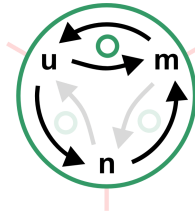
Figure 8
system archetypes
(with real-world
examples)



performance-driven

Flow is possible through “direct involvement” with the *music* via *performance*, which can be *auditioned*, or *transcribed* for *visualisation*. The system harnesses *performance* skills without requiring (or benefiting from) *data manipulation* skill or literacy with *notation*.

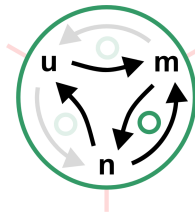
e.g. A **sequencer or digital audio workstation (DAW)** enables direct interaction with music through a MIDI or acoustic instruments, from which the output is recorded and visualised.



audition-driven

Flow is possible through “direct involvement” with the *music*, via *performance*, but the data is manually entered, in *structured composition*. The system allows the *user* to “jam” or practise before committing to an idea, which is entered manually and auditioned. The *user* manually inputs music in a format acceptable to the notation.

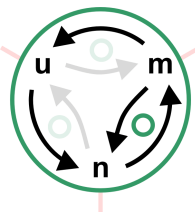
e.g. **Augmented instruments** focus interaction on live performance through a union of acoustic sound and electronic processing, the latter of can be manipulated using a visual UI (e.g. a computer or smartphone).



transcription-driven

Flow is possible through the *recording* macro-process, but performance relies on visual, not aural feedback. Acoustic feedback can be present, but is less important. The estrangement of the *transcription loop*, makes understanding complex mappings between *notation* and *music* challenging, but this could be the intention.

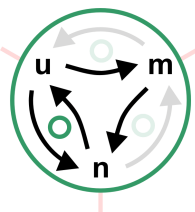
e.g. **Guitar tuners** and **vocal trainers** provide visual feedback to augment or replace audio feedback, making it easier to discern subtle variations in pitch. Interaction in music games, such as **Guitar Hero**, is also driven by simplified visual cues and actions, only loosely coupled to musical playback.



realisation-driven

As above, the *user* is estranged from the *transcription loop*, but flow is possible via the *composition* macro-process, driven by *data manipulation*. The lack of visual feedback challenges the user to infer the mapping between notation and music using only *audible* feedback.

e.g. **Max/MSP** enables complex mappings between user input (*gesture* and other non-visual modes) and intricate sound and musical processes. **Active listening** similarly involves an indirect coupling of control and digital music playback.

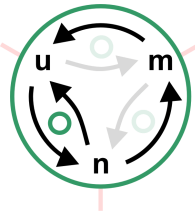


visualisation-driven

Flow is supported between the *user* and the *notation*, and also in the *recording* macro-process. Data can be *manipulated*, or *transcribed* from a *performance*, for *visualisation*. Such systems are concerned with visual representations of music, rather than the music itself.

e.g. Through visualisation, systems for **performance analysis** reveal details about musicianship that performers cannot articulate. In **score editors**, composers use performances to enter data with the goal of typesetting notation.

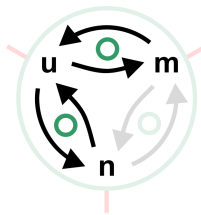
Figure 8 (contd.)
system archetypes
(with real-world
examples)



manipulation-driven

Flow is supported between the *user* and the *notation*, and also in the *composition* macro-process. Users *manipulate* music through the *notation*, but receive *visual* and *aural* feedback. Thus, whilst the *notation* determines what is musically possible, the result is heard in non-abstract terms, helping the user understand how what they see in the notation relates to what they hear in the music.

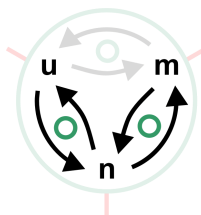
*e.g. Notation-oriented programs emphasising rapid feedback like **trackers** (Section 2.1) and **live coding environments** like SuperCollider and Max/MSP (when focus is on the UI). When used for composition rather than transcription, **score editors** also provide audio feedback to guide interaction with notation.*



user-mediated

In this *symbol-based modelling* scenario (Leman, 2007), flow is possible between *user* and *music*, as well as *user* and *notation*, but the *user* is responsible for determining the mapping between *notation* and *music*. Focus and concentration are split, and the challenge requires skills in two distinct areas, *performance* and *manipulation*.

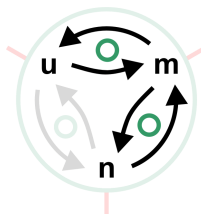
*e.g. The lack of transcription or storage in acoustic instruments and **analog synthesisers** requires composers to transcribe their music using an alternative method of notation. Uncoupled to the sound source, such methods (e.g. **sketching on paper**) may not provide musical feedback, placing demands on literacy.*



notation-mediated

The *user* interacts with the *notation* visually, and the *notation* interacts with the *music*. The *user* achieves flow, but is estranged from the *music*, and the notation's designer (the programmer) determines the mapping of *notation* and *music*.

*e.g. In **algorithmic composition**, artists focus on abstract representations of musical processes. In digital music, low liveness systems with delayed musical feedback (i.e. compiled languages, such as **CSound**) are effectively visually-mediated.*



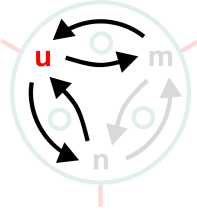
music-mediated

*(e.g. **gesture**, Max/MSP, audio recording)*

Flow is possible through “direct involvement” with the *music*, via *performance*, which goes through automatic *transcription* to data in the *notation*. Users are estranged from the data, and are able to record and playback the musical data, but not effectively *manipulate* it.

*e.g. Many automated processes can act on the sound or music (MIDI) during a live performance without user intervention, from conventional DSP **effects processing** (reverb, echo, EQ, etc.) to more advanced uses of music programming languages (e.g. **Max/MSP** or **SuperCollider**). Such processes may also be affected by other users, which may be modelled by adding additional user nodes to the network, or by integrating other system archetypes (e.g. sharing the transcription loop with another user's manipulation-driven system).*

Figure 9
system features

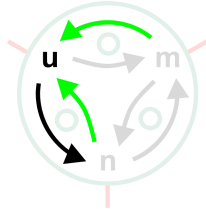


flow congestion

The possible paths through conceptual spaces increase exponentially as more sub-processes are involved, complicating the system design or user experience. *e.g. systems with multiple notations (see Figure 10a) or user-mediated systems (see Figure 8)*

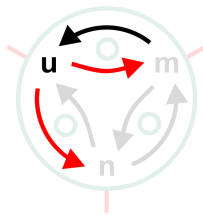
flow redundancy

Conceptual spaces fed by more than one sub-process can increase the opportunities for maintaining flow, at the cost of complicating processes in the conceptual space. *e.g. the combination of visual and aural feedback in manipulation-driven system like trackers (see Figure 10b)*



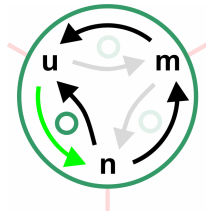
flow fission

Conceptual spaces that feed more than one sub-process, potentially divide or redirect flow and focus, which can positively (or negatively) impact the user experience, depending on context. *e.g. systems with multiple notations or input methods (e.g. digital audio workstations, Figure 10a)*



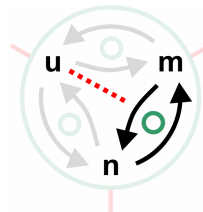
flow interference

Systems that mix overlapping feedback loops (i.e. an intrinsic micro- and macro-process) combine *flow redundancy* and *flow fission*; potentially combining the impetus of each in a way that can reinforce system flow. *e.g. in trackers, visual and audio feedback respectively feed interaction cycles with the notation and music (Figure 10b)*



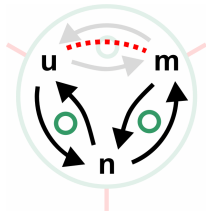
flow estrangement

All intrinsic micro-processes lie apart from an opposing conceptual space (e.g. user vs. transcription loop). From the user's perspective, any such estrangement can obfuscate system behaviour, making it difficult to learn and predict. *e.g. realisation- or transcription-driven systems*



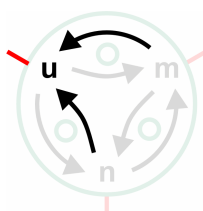
system indirection

Systems that channel flow around or through a single conceptual space can, respectively, divide or complicate the interaction. The mappings between conceptual spaces (e.g. user and domain) become less clear, and interaction becomes less direct. *e.g. notation-mediated systems offering "indirect involvement" in music (Leman, 2008)*



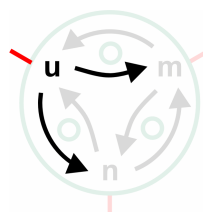
extrinsic reward

Where other users or external sources of oversight exist in the user experience, the system becomes dependent on *extrinsic* factors, making the user self-conscious and impeding flow. An *extrinsic reward* often also indicates an *extrinsic motivation* (see below). *e.g. other agents, such as collaborators or audiences in social situations (Section 3.7)*



extrinsic motivation

Where the impetus for action is derived from an external source, interaction depends on stimuli outside the user's or system's control. While this can aid novices needing extra guidance, it harms their sense of autonomy and can thus impede flow. *e.g. a tutor, instructor, supervisor, collaborator, or even documentation*



4.2.4 modelling ‘liveness’ in a musical system

The model presented in the previous section lays out a foundation for representing the creative user experience as a network of closed feedback loops. Figure 9 looked at specific features of the network that impact the interaction, relating to formation of loops in the graph. However, it is also critical to consider the quality of the feedback, when such loops are present in a system.

*Tanimoto’s
levels of ‘liveness’*

Tanimoto (1990) introduces the concept of “liveness” to the practice of programming, unifying notation and interaction by considering the different ways the end product (an executable program) is manifest during the development process, and how manoeuvres in the notation affect the resulting execution of code. As described in Table 4, he identifies four increasing levels of liveness, each offering a decreasingly-abstract picture of execution.

In contrast to other uses of the term “liveness” in the performing arts (Auslander, 1999; Emmerson, 2007; see also Footnote 3, p90), Tanimoto’s HCI concept of liveness is applied here for its capacity to describe single-user systems and relationships between user and notation, and its clear delineation of specific levels and properties affecting liveness in the user experience. An interesting question for future research, however, is whether a similar operationalisation of liveness could be applied to Auslander’s discussions of issues in performing arts, i.e. levels of performance liveness.

Table 4 (below)
Tanimoto’s four
levels of liveness
(from Church, Nash
and Blackwell, 2010;
and with Tanimoto’s
original descriptions)

In programming, Level 4 liveness is rare, since it places constraints on the power of *useful* abstraction available to the programmer. Blackwell (2002) highlights such abstraction as a critical programming tool that governs the possible complexity of programs that can be encoded by a notation.

Level 1 liveness (*informative; “ancillary”*)

describes situations in which a visual representation is used as an aid to software design (Tanimoto was referring to a user document such as a flowchart, not a programming language). This provides a basic level of graphical representation, and can be made continuously visible, although mainly because of the fact that a paper document can be placed beside the screen, rather than on it.

Level 2 liveness (*informative, significant; “executable”*)

describes situations in which the system can use the visual representation as an executable specification (i.e. a visual programming language, but only as offering graphical input to the compiler, rather than being continuously interpreted). This provides a basic kind of physical action, in that modification of the representation will eventually change the program’s behaviour.

Level 3 liveness (*informative, significant, responsive; “edit-triggered”*)

describes situations in which the representation responds with immediate user feedback, for example via interactive syntax checking. This allows users to make rapid actions, and often (after noting the system response) an opportunity to reverse an action that was incorrect.

Level 4 liveness (*informative, significant, responsive, live; “stream-driven”*)

describes situations in which the environment is continually active, showing the results of program execution as changes are made to the program. This provides visibility of the effect of actions.

recording in
programming

Some consumer programs, such as *Microsoft Office*, allow end-users to record interaction, as an alternative to coding automation in a scripting language (e.g. VBA) – the application automatically generates the code to reproduce the original interaction, which can then be executed, viewed and edited. Such *macro recording* may be seen as implicit Level 4 liveness, since the process engendered by the final code mimics precisely the actions used to generate it. However, it also exemplifies the expressive power lost through the lack of opportunities to abstract processes across time (e.g. iterative loops) or context (e.g. conditional statements).

music as
programming

Church, Nash and Blackwell (2010) use music as an analogy to programming, to illustrate the similar trade-off in the recording of a musical performance. Using a MIDI or acoustic instrument to record a ‘live’ musical performance, in realtime, allows “direct involvement” in music (Leman, 2008), during which the effects of actions (on the instrument) are continuously and immediately *audible*. Sequencers are unable to sustain Level 4 liveness after the point at which the performance is captured, instead providing sub-devices (e.g. Arrange Window, Score Editor, Piano Roll, etc.) each allowing the visualisation and editing of specific and distinct aspects of the recorded data, where interaction is driven by visual feedback, and less frequently auditioned by spooling to the appropriate point and initiating playback. This subsequent process of transcribing, abstracting and editing the result, as in a DAW, significantly lowers the directness and liveness of interaction.

liveness in music

Table 5 gives programming and musical examples that conform to each level of liveness. Liveness is a property of the notation and the user experience (Church, Nash, and Blackwell, 2010), and varies between both interfaces and users, depending on the interface’s implementation and user background, respectively. These examples are only offered as a guide, to expose general trends and factors in common interaction styles; specific programs may be more (or less) susceptible to liveness issues, or mitigate issues in one part of the program with the provision of another. Indeed, the variety of tools and UI styles in a DAW can be seen as an implicit attempt to tackle the apparent trade-off between liveness and abstraction power.

<p>Table 5 – examples of the each liveness level, in both programming and music</p> <p>(based on Church, Nash and Blackwell, 2010)</p>	liveness	in programming...	in music...
	1	flow chart, UML diagram	composer shorthand, arrange window
	2	code editor, compiler	score, data list, piano roll, CSound, OpenMusic
	3	code completion, syntax highlighting, edit & continue	soundtracking, live coding
	4	macro recording	sequencer/DAW recording, live performance, mixing

At the lowest end of the scale, Level 1 liveness can be seen in the offline notations used by composers, such as the sketching of ideas and musical processes on paper. The macro- and meta-editing aspects of musical arrangement, such as the visual delineation and decontextualisation of musical phrases, repeats, forms, and structures represent highly-abstract levels of musical notations that are only tacit⁹ in the final sound. As such, many of the functions of a sequencer's *Arrange Window* might be seen in this context – part boundaries, sections, markers, labels, colours. The central position of this view affords a top-down composition process, contrasting the bottom-up process afforded by the wider sequencer user experience – that of recording the individual notes and musical events through performance capture. In modern packages (e.g. DAWs), direct manipulation techniques, such as *drag-'n'-drop* and related clipboard operations allow macroscopic editing that affect the musical output and improve the liveness of the *arrange window*, but still rely on other interaction techniques to permit the entering and editing of individual notes and musical events.

*Level 2 and 3 liveness
in musical scenarios*

The majority of other computer music scenarios are centred on editing a visual specification of what will happen in the music, as in both a sequencer/DAW's GUI and trackers. Such programs thus lie somewhere on a continuum between Level 2 and Level 3 liveness, based on the immediacy and quality of feedback provided. In most modern music programs, the music encapsulated in a visualisation can be interpreted and realised (executed) in realtime, at a quality approaching that of the final master copy – thus presenting no technological impediment to liveness; which becomes an issue for the interaction designer. One factor that determines the perceived liveness of an editing episode is the delay between the editing action and the auditioning of the result – in Level 2 liveness, the user completes an edit, then manually triggers an update; in Level 3, the update is triggered by the edit automatically.

*Level 3 liveness
in live coding*

In *live coding*, programming languages like *SuperCollider* and *ChucK* have modes where music or audio source code is subject to *automatic interpretation* (Blackwell and Collins, 2005), often used for editing a live performance. The incremental edit-and-update style favours progressive, generative, or textural musical styles (a “code and run” aesthetic – *ibid.*) that focuses on processes, rather than events – and where it can be hard to address individual notes. Such Level 3 liveness, however, greatly improves the directness of interaction, in contrast to the Level 2 liveness of older, non-realtime languages such as *CSound* and *OpenMusic*.

⁹ Literally, from the Latin, *tacitus*, meaning “to be silent”. In music, *tacet* similarly denotes silence.

In mainstream music programs, such instant, automatic feedback is only practical for very simple edits. In many programs, for example, entering or selecting a single note will trigger playback of that isolated note. For more involved edits, the user must retain control of how the wider musical picture is presented (e.g. through playback controls on the *transport bar*), or otherwise risk a fatiguing cacophony of sound (arising from the program's relentless playback of even the most trivial changes) or the perceived loss of control and transparency if the program itself triages edits for playback.

*controlling playback
in a sequencer/DAW*

When sonic feedback is actively managed by the user, the program supports only Level 2 liveness, and it becomes the user's responsibility not only to choose when to trigger playback, but to select what should be played back. In sequencers and DAWs, playback is controlled using the *transport bar*, using the metaphor of a *tape recorder*, with play, record, pause, fast-forward and rewind buttons. As such, the program also maintains a separate cursor for playback (*song position*), independent of other editing cursors or focus. Consequently, when a user finishes editing a part of the music, it is necessary for them to align (spool) the playback cursor to the edited section, before the edit can be auditioned. To accelerate the process, many programs allow *markers* to bookmark common points in the piece, and associate them with specific shortcut keys. However, managing bookmarks requires foresight and planning, representing both a *premature commitment* and an *attention investment* (Blackwell, 2002), reducing the dynamism of the creative user experience. The cost of auditioning the music encourages the user to audition the music less frequently, and refocuses their attention on the notation and visual feedback, rather than more concrete aural feedback (see Chapter 8).

*controlling playback
in a soundtracker*

By contrast, the keyboard bias of soundtrackers requires them to constantly maintain a single editing focus, to indicate which part of the notation keyboard input will be directed. Playback of the song can be easily triggered from wherever this editing cursor lies within the music, using a single key press (e.g. F7). During playback, the editing cursor can also be slaved to the playback position, to ensure the visual and aural focus remain aligned. In such cases, the editing cursor functions like a sequencer's playback cursor, enabling similar realtime (live) music entry and editing. At other times, the visual and aural focus remain closely linked, and the reduced cost of auditioning edits encourages the user to rely more on the aural feedback, rather than the visual representation in the notation.

The architecture of the tracker song similarly encourages tighter feedback cycles. The user's focus is narrowed by the division of the

song into a sequence of short musical phrases (*patterns*, typically 4 bars in length), the start of which presents another point from which playback can be triggered (Ctrl-F6), using a single key press, and over the length of which playback can be looped (F6). This allows the user to listen to a recent edit with the appropriate musical pre- and post-ambles, to cheaply audition it in the local musical context, without having to move a cursor. The entire song can be similarly played back from the start, using a single key press (F5). Naturally, it is just as simple to stop any current playback (F8).

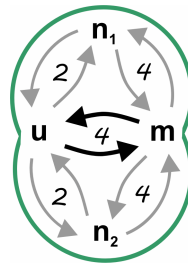
In trackers, liveness is improved by attenuating both the delay and effort involved in requesting aural feedback. Moreover, the physical actions can be learnt, so that the user triggers aural feedback after an edit reflexively; cognitively, it becomes *automatic* to hit F6 or F7, to hear what it sounds like. Under such conditions, the tracker user experience supports Level 3 liveness (see Chapters 7 and 8).

In Church, Nash and Blackwell (2010), Tanimoto's concept of liveness was combined with the *systems of musical flow* model (outlined earlier, in 4.2.3), by annotating the constituent feedback loops with their corresponding level of liveness. Figure 10 shows the two musical scenarios discussed in previous paragraphs.

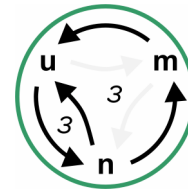
liveness and flow

Figure 10
Feedback loops in two
music program types,
annotated with
levels of liveness

(from Church, Nash
and Blackwell, 2010)



(a) digital audio workstation (DAW)

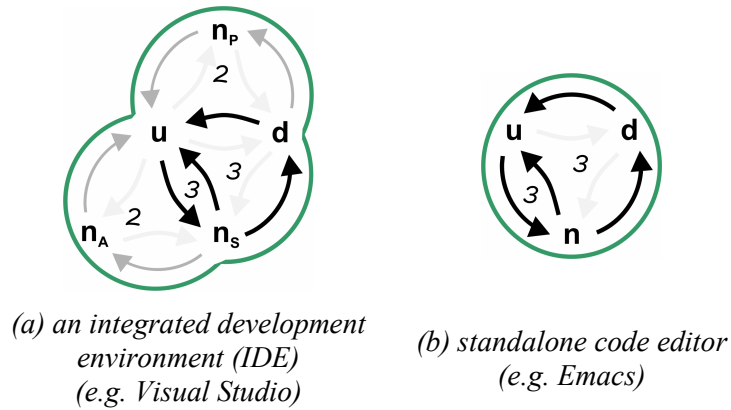


(b) soundtracker

Figure 10 (a) illustrates the divided sequencer/DAW user experience: a *performance loop*, straddled by notations representing two of the common DAW editing 'devices' (see 2.1). Interaction centres on capturing a performance, the notation-less process of which implicitly supports Level 4 liveness. Outside performance, multiple visual notations compete to present different aspects of the musical recording, splitting interaction between different views or windows, each allowing editing with Level 2 liveness, focusing the interaction on visual, rather than aural feedback. By contrast, the user experience of the soundtracker is illustrated in Figure 10 (b) – the *manipulation loop*, augmented by feedback from the domain; as in the basic *manipulation-driven* system archetype (see Figure 8). The tracker prioritises a central notation, and supports rapid sound feedback to support higher (Level 3) liveness during editing.

Church, Nash and Blackwell (2010) likened the soundtracker user experience to that of standalone code editors (like *Emacs*), both favouring keyboard input, text output and expert use. Indeed, the similar interaction modalities and audiences engender equivalent flow schematics, as shown in Figures 10 (b) and 11 (b) respectively.

Figure 11
Two programming experiences, modelled using feedback loops, with *source code* (n/n_s), and other notations (n_p , a *code profiler*; and n_A , a *static analyser*)
(from Church, Nash and Blackwell, 2010)



The article likewise drew an analogy between *integrated development environments (IDEs)* and the integrated musical production environment of the DAW. However, here a distinction is evident; as IDEs retain the *code editor* as a central, primary notation, extending the system's functionality with ancillary notations (e.g. code profiler, static analyser) that remain in the periphery of interaction, as illustrated in Figure 11 (a). As such, this might suggest a future direction for DAWs, were one of the existing sub-devices to be extended and given greater prominence in the user experience.



chapter five **iMPULS: internet music program user logging system**

This chapter outlines the structure of an investigation into virtuosity and flow in computer music interaction, focusing on the analysis of interaction logs from a large number of users of *reViSiT* (Nash, 2004), an established tracker program, as well as sequencer/DAW packages.

In the experiment, participants download and run the software on their own computers, using it to write music. As they use the software, information about their interaction with the program is gathered. When the program is closed, the recorded data is sent to an Internet server, for collation and analysis. This chapter details the workings, development, preparation, and running of the system, ahead of subsequent chapters, where the experiment's results and findings are presented.

As established in Section 3.4, creativity research presents a methodological dilemma. Controlled experiments seek to observe or measure specific cognitive processes under controlled or constrained conditions, which lie at odds with the freedom or autonomy that psychologists and artists recognise as crucial to many forms of creativity (Sternberg, 1999).

Other investigations focus on reviewing the biographies and repertoire of the “creative genius”, which often relies on subjective, introspective, and fragmented accounts of the subject’s motivations and experiences, whilst also restricting the type of creativity to that recognised by society, historically (*H-creativity*) – rather than the individual, personally (*P-creativity*) (Boden, 2004). With the rise of the Internet and online end-user communities, subcultures have appeared, where audience tastes are more specialised, feedback is less critical, and the threshold for entry and recognition is lower. As a result, home users are increasingly becoming the target audience for music software developers (e.g. *Band in a Box*, *Guitar Hero*), whose focus is shifting away from career music production professionals – and, at the same time, from creative end-product to user experience.

The goal of our experiment is to objectively investigate creativity “in the wild”, and investigate the role of interface design in the creative user experience, looking empirically:

- for **evidence of virtuosity**, where users have developed skills enabling them to tackle more challenging or complex creative tasks; establishing UI factors supporting (or hindering) learning.
- for **evidence of flow**, where users demonstrate an ability to stay focused and engaged with a task, as virtuosity is developed; establishing UI factors supporting (or hindering) flow.
- at **end-user creativity**, not just creative geniuses, where a user’s attainment is measured personally, and the user experience drives itself, rather than the promise of an end result or prize; where the task is *intrinsically-rewarding*.
- at a **private, uncontrolled setting**, where the user is free to experiment in their own space, with their own PC and software, in their own time, in their own way; where ego, self-doubt and *extrinsic factors* are less involved.
- to **develop quantitative techniques** for evaluating user experiences and interfaces, in creative authoring software, that can be efficiently and economically used to highlight issues with virtuosity, flow and user experience.

This section introduces the *reViSiT* software, the platform used in later analyses of virtuosity and flow. As an example of tracker software, *reViSiT* provides the opportunity to scrutinise a user experience, in which users have observed many hallmarks of ‘flow’ – concerning focus and concentration, skill development, action-awareness merging and even a distorted sense of time.

These next pages describe the software and its user community, and how both were prepared for the ensuing investigation.

5.2.1. background

With many of the established tracking packages based in *DOS*, the popularity of tracking waned significantly with the advent of *Windows XP* in 2001. Popular tracker programs, such as *Fast Tracker 2 (FT2)* and *Impulse Tracker 2 (IT2)* could no longer run in this new environment. At the same time, the rise of the desktop studio and the increasing power of home computers encouraged manufacturers to move from hardware-based DSP to software. Tracker users migrated to soundcards and sequencers compatible with new innovations, such as software synthesizers and effects (e.g. VST plugins) and high-quality, low-latency audio drivers (e.g. *ASIO*, *WDM*), unavailable in most trackers.

reViSiT was originally developed as “VSTrack” (Nash, 2004), an academic project to resurrect the tracker user experience and integrate it with that of sequencers, allowing tracker musicians to take advantage of modern music technologies and provide sequencer users access to the benefits of the tracker user experience. The project endeavoured to offer the best of both worlds, wherein a composer might enter music that is suited to a MIDI or acoustic performance using the sequencer’s recording process, but switch to the tracker’s notation-mediated interface, for musical edits requiring a more flexible interaction cycle – such as drum or synthesizer programming.

5.2.2. program overview

Pictured in Figure 1, *reViSiT* unifies tracking and sequencing by presenting a tracker interface as a *VST Instrument (VSTi) plugin*, which can be loaded into any compatible VST host (e.g. a sequencer).¹ Unlike most VSTi plugins, input comes not from realtime MIDI messages, but from asynchronous (“offline”) computer keyboard interaction. Using the tracker interface, the

¹ *VST (virtual studio technology)* is an industry-standard plugin format, developed by Steinberg, for hosting software-based effects and synthesizers in compatible music programs.

user works on a parallel tracker song, which is rendered and relayed to the host for audio output or further processing, as needed. The plugin maintains its own playback and edit cursor, allowing the user to freely audition any part of the tracked music during editing, but automatically synchronises playback, when it is triggered from the host. Optionally, the edit cursor can also be synchronised to the host, to allow realtime musical input to the tracker, during playback.

VST plugin architecture

Functioning as a plugin, the hardware layer is abstracted; *reViSiT* simply populates an audio buffer, which becomes the responsibility of the host to handle further. As such, *reViSiT* automatically works with any audio devices and protocols supported by the host program, such as *ASIO*, *WDM*, *DirectX*, etc. Similarly, the host's support for VST means that any audio or MIDI output from the plugin can be connected with other VST plugins, allowing *reViSiT* users to avail themselves of the host's various software synthesizers, effects, and hardware connections, from their tracker song.

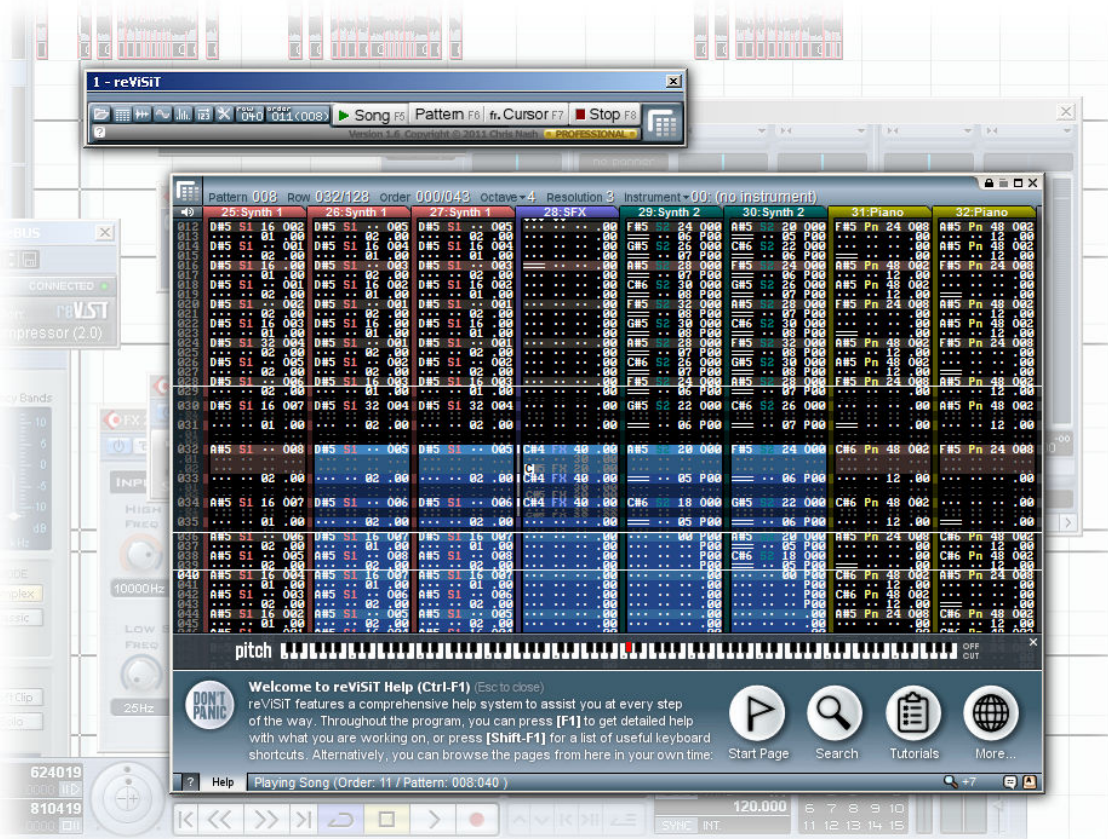


Figure 1 – the *reViSiT* tracker plugin, comprising a main editor window (showing the pattern editor) and toolbar (which simply acts as a placeholder or anchor, within the plugin host environment; see Section 5.2.4). The UI is based on that of *Impulse Tracker 2* (see Section 2.2 for a detailed description of the program and notation) – a number of extensions to which are visible in the figure, including colour-coded instruments and channels, high-definition sub-row editing, context-sensitive graphical editing guidance and feedback (e.g. pitch represented as a piano keyboard), and an integrated help system.

Figure 2
reViSiT (right) running
 under *Cubase SX* (left)



*tracker heritage
 and compatibility*

reViSiT's interface and notation draws heavily from *Impulse Tracker 2* (*IT2*, see Section 2.2.1); user input is almost exclusively through the keyboard, and both data and UI objects are almost exclusively represented as text. The program supports the importing of older tracker formats, including MOD (Amiga), XM (*FT2*), IT (*IT2*) and S3M (*ST3*), but its expanded feature set can only be saved in its native format, which simply packages together an XML description of the music with the samples used (in WAV format) in a compressed ZIP archive. This data is automatically embedded in documents saved by the host application, but can also be exported to a separate file, to facilitate open interchange of music and samples with other programs and users.

target audience

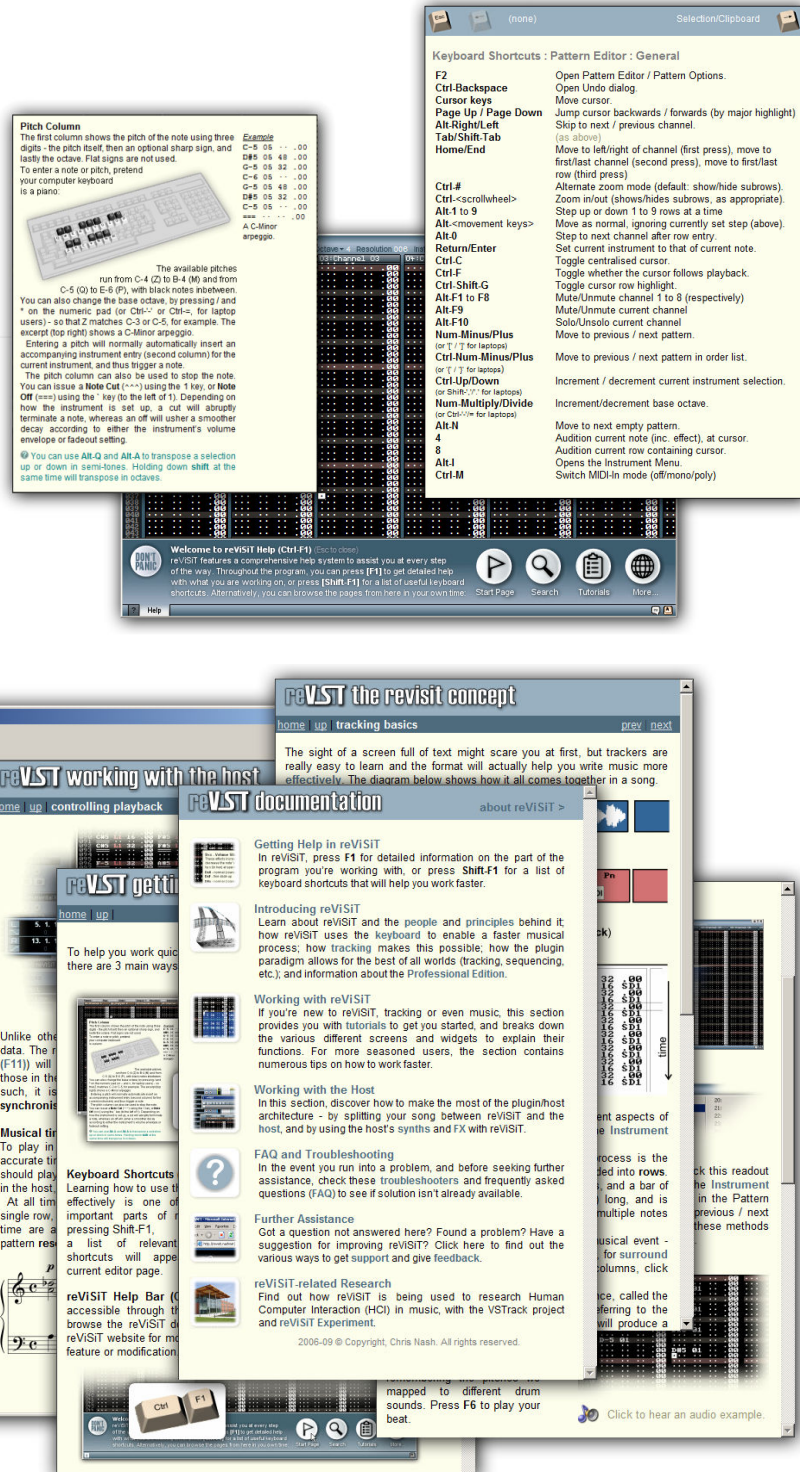
reViSiT provides a natural upgrade path for users of older trackers (notably *IT2* and *ST3*), remaining faithful to their use of terminology and keyboard shortcuts, allowing users to draw on knowledge and skills learnt previously.

At the same time, the project aims to attract new users and develop the user experience. For musicians and sequencer users, the learning requirement associated with tracking software represents a large *attention investment* (Blackwell, 2002), which discourages users from even engaging with the technology. As a plugin, the investment is reduced, allowing new users to experience tracking without leaving their existing music software environment. At the same time, *reViSiT* includes a tightly-integrated support system, providing popup, context-sensitive crib sheets in many parts of the program to reduce the onus on memory, in addition to visually-rich help and tutorial pages (see Figure 3) that encourage learning-by-doing and experimentation.

Figure 3 - reViSiT's integrated support documentation

(top) context-sensitive popup tooltips explaining notation usage and syntax (pitch column shown, left) or providing reference lists such as effects or keyboard shortcuts (shown right)

(bottom) integrated *HTML Help* documentation, with more detailed guides, tips, tutorials, and descriptions, and pages also linked with current program focus.



5.2.3. reception and user community

reViSiT was released as closed-source freeware, and quickly became popular with users and reviewers, earning a place in *Computer Music* magazine's *Freeware Top 50* awards; described as “perfect for those who need both audio-handling power of the modern DAW and the quick, hands-on, detailed editing that trackers provide.” (Robinson, 2007)

user backgrounds

A short survey presented to users before they downloaded beta versions of the software produced 4,981 responses and suggests the plugin format is successful in attracting both former tracker users and more traditional musicians without tracking or advanced computing experience:

- 71.9% (3,583) stated proficiency in **music**
- 41.5% (2,066) stated proficiency in **tracker**
- 38.7% (1,926) stated proficiency in **programming**
- 27.8% (1,339) stated proficiency in **music only**
- 19.8% (974) stated proficiency in **tracking & programming**

These figures only record downloads, and do not necessarily reflect the backgrounds of users that go on to use *reViSiT* on a regular basis. Indeed, other user feedback suggested that those with tracker experience were more likely to continue with the program, and that novices were perhaps still intimidated by the tracking environment. This reinforces the value of the plugin format, by lowering the barrier to trialling the software and allowing analysis of the pivotal initial moments and obstacles for users that do not continue beyond the first few minutes.

forum and user community

Users that persevered have formed an active online community, centred on the *reViSiT Forum*.² The forum acts as the official news source of the project – new releases, development updates, and related topics. Over 350 discussions (almost 2000 posts in total)

cover a variety of subjects; chatting, posting comments, seeking help, reporting problems, suggesting changes or features, and exchanging music tracks (or even videos) made with *reViSiT*. As a result, even though the source code is closed, individual users have a large and perceptible influence on development of the program and interface, wherein design decisions are proposed, then discussed and debated by users, before being put to code.

Within the online community, a handful of more enthusiastic users³ make greater contributions to the project: maintaining a presence on the forum, staying abreast of project developments, providing detailed test reports of new versions, and sharing their knowledge and music with novice users and other visitors. Indeed, it is the openness and enthusiasm of *reViSiT*'s user community that have enabled this study of the program's real-world use.

Figure 4
(front to back)
reViSiT forum,
website and FAQ



² <http://forum.nashnet.co.uk>

³ See Acknowledgements (p5).

5.2.4. development and testing

To gather as much data as possible, from as many individuals as possible, the study has to attract people to the software and also ensure users are not deterred by other factors.

*data privacy
concerns*

A limited number of users will join the experiment out of enthusiasm for the research or loyalty to the developer, but most will be more circumspect about surrendering their data and privacy, especially in light of the recent blight of spyware and attempts by both hackers and companies to gather user data. Ethical approval and the mark of the *University of Cambridge* will allay some fears, but provide little incentive for users to update to a special version of free software that collects user data.

*participation
incentives*

Like many experiments involving human volunteers, a more tangible incentive is required. For small studies; sweets, vouchers or money are often used, but the size and spread of our sample makes such offerings impractical. An alternative is to make the experiment the reward itself, offering an experience that individuals would otherwise find difficult or too expensive; the promise of money saved, rather than received.

quid pro reViSiT

Since a version of *reViSiT* was already freely available, there was little to entice current users to take part. However, prior to the research, a “*Professional Edition*” of the software had been mooted on the forum, promising a significantly expanded feature set, tailored and priced for professional users and enthusiasts. Consequently, it was decided to develop this edition of the program to give away to experiment volunteers. Perceived as a monetary reward, it provides an incentive to take part in the research, as well as compensation to offset privacy concerns.

Figure 5 lists the four main features exclusive to the *reViSiT Professional*, not only designed to extend the free edition, but representing innovations not generally found in other programs, designed to broaden the appeal of the program to musicians working with video, audio, MIDI, and trackers, respectively.

Figure 5
exclusive features in
reViSiT Professional

surround sound support – extending stereo output to 5.1 channels, with special notation syntax and pattern effects to address extra panning options (e.g. Cartesian or polar coords, depth and rotation slides, discrete channels).

advanced output routing – enabling different notes, samples, instruments, or channels to be sent to the host on separate audio outputs (for effects and post-processing), where routings can be set *on-the-fly* using pattern effects.

MIDI-triggered patterns – allowing users to control the order of playback via MIDI, triggering patterns using notes from a MIDI device or recorded track, enabling live arrangement or editing via the host’s visual editors.

hi-res timing & hi-def editing – addressing the perceived rigidity and low timing resolution of trackers by enabling users to ‘zoom into’ or ‘open-up’ the pattern grid, placing musical events between rows (see Figure 1).

reducing the impact
of program errors

Once registered, a user is solely responsible for the running of the program. Whereas lab-based participants may feel morally-obliged to complete the prescribed procedure and not walkout in the middle, there is little to inhibit our user from abandoning the experiment, by simply closing and deleting the software.

Program errors are especially likely to discourage and deter participants. Whereas bugs and oversights are more easily tolerated in free or academic software, users will demand a complete, fully-tested, and reliable program if it is billed as a commercial product. Moreover, errors in the program may also interfere with user interaction, where unanticipated program behaviour risks impeding the very flow sought by the experiment. As such, significant effort was invested in the testing of the *reViSiT Pro*, and the code to support the experiment. Figure 6 summarises the various milestones in the development and testing of *reViSiT*.

user feedback
and testing

The limited resources of lone developers are often most keenly felt in aspects of quality assurance. The user interaction and audio engine in *reViSiT* is complex, especially compared to more conventional effect and instrument plugins, and whilst the new features were quickly coded, debugging the complete program took significantly longer.

Defensive coding and unit testing helps reduce bugs that appear in individual lines and sections of code. Object-oriented, interface-driven coding techniques also help simplify the integration of different system components. However, it can be hard to anticipate how a system will be used, and thus many issues only present during use. The problem is arguably exacerbated for developers of authoring tools, which seek to support flexibility and creative freedom, and where original and unforeseen uses must be expected.

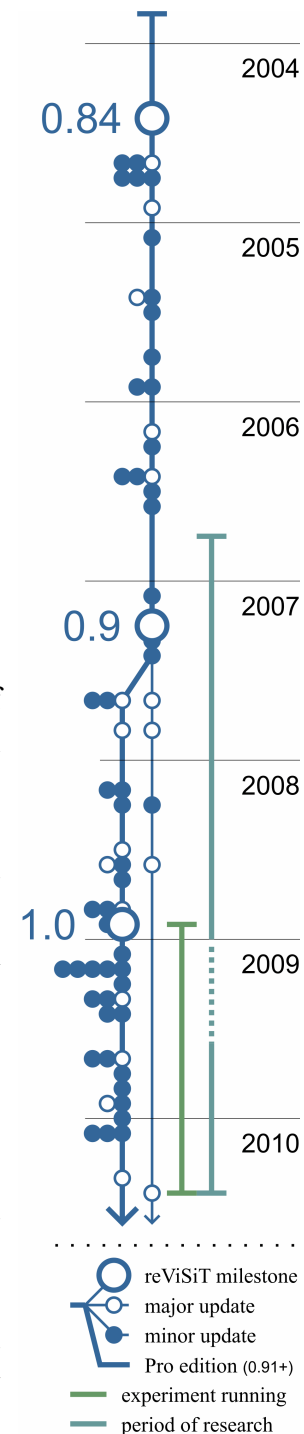


Figure 6
development timeline

*VST plugin
keyboard support*

Moreover, the plugin architecture presents extra debugging challenges. The *VST plugin specification* (Steinberg, 1999) tries to standardise the communication and interactions between plugin and host. However, manufacturers have different approaches and styles, and thus host behaviour varies. Many hosts stray from the specification and others implement only part of it, requiring plugin developers to extensively test plugins for compatibility problems.

For example, many manufacturers assume that plugins will be manipulated using the mouse, and thus provide little or no support for keyboard input. The VST specification contains no mechanism for hosts to tell plugins what, if any, keyboard support they provide. Even where support exists, many hosts relay only a subset of keystrokes to plugins, holding back others for their own use, regardless of whether the plugin has focus or not.⁴

*multi-threading
and thread safety*

Another pitfall for developers has emerged relatively recently, and concerns the increasing use of multi-threading. As a plugin, *reViSiT* exists as a dynamic-link library (dll), a set of routines that run in the threading environment of the host program. Thread-safety problems can arise when functions are called concurrently and use shared resources (e.g. memory). Modern VST hosts run high-priority threads for audio processing in parallel with lower-priority threads for the UI, meaning users create, delete and change musical data that is, at the same time, in use by the audio engine. In the worst case, delayed execution leads to glitches or ‘dropouts’ in the sound output (for the audio thread) or stuttering and poor responsiveness in the interface (for the UI thread).

Newer versions of *reViSiT* use mutex objects to guard against such conflicts, where threads wait until it is safe to access an object. At the same time, identifying and debugging concurrency issues can be problematic and time consuming. Unlike some code errors, memory and resource conflicts can be hard to reproduce, and may only cause problems in a tiny fraction of use cases – when multiple threads are unfortunate enough to contend for the same object at the same moment.⁵

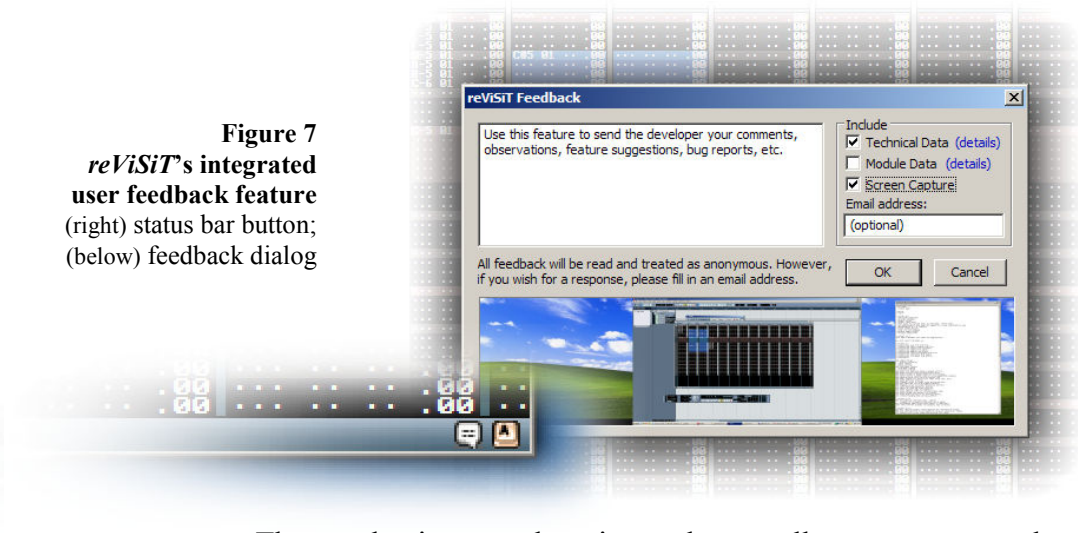
⁴ In tracking, keyboard support is paramount and, despite trying various workarounds (e.g. keyboard hooks and other Win32 ‘hacks’), no solution ensures uncensored keyboard input reaches the plugin window in every host. In many cases, the host’s parent window intercepts, filters, or blocks keystrokes, allowing users to click in the plugin, but not control it by keyboard. In what became an undesirable but necessary compromise, *reViSiT*’s UI was split into two windows: the standard VST plugin window becomes a simple, mouse-controlled transport/tool-bar, created and positioned by the host; the main editing environment resides in a separate window spawned by the plugin itself. This window is not owned by the host, and thus receives unfiltered keyboard input from the OS directly. The impact of the split UI is minimised by making the toolbar functionality superfluous, duplicated in the editor window.

⁵ For example, if two threads access a shared resource roughly once a minute, and that access takes 10ms, the chance of a collision, in a given minute, could be as low as 0.000003% – the code could run continually for 8 months before an overlap occurs, which may not even then lead to an error.

As such, it is not possible for developers to test all eventualities, and the burden must be shared between a wider circle of testers, using pre-release alpha and beta versions of the software, and across different computers, hosts, and interaction styles.

Although some users have programming experience, many come from a non-technical, musical background and are unable to give technical details when issues occur; many others were more interested in simply using the program, rather than helping with testing, and don't bother to email about problems they encounter. To address this, *reViSiT* 0.92.1 introduced an automated feedback system (Figure 7), allowing the program to easily send user feedback directly back to the developer, along with technical data describing the fault and code location.

Figure 7
reViSiT's integrated
user feedback feature
(right) status bar button;
(below) feedback dialog



The mechanism can be triggered manually, to comment about the program or report a *soft failure* (unexpected, but non-catastrophic behaviour, not prompting an explicit error message). In the event of a *hard failure* (an unhandled exception or program crash), the mechanism activates automatically. In both cases, a dialog appears, asking to send technical information (a stack dump, with system and error details) and also asking the user for any additional information they can provide, including a message (error description, steps to reproduce, recent system activity, etc.), screenshot, or copy of the music data being edited.

The feature has been invaluable in fostering user feedback, and accelerating the debugging process. The stack dump can be used to recreate the crash, using a debugger, which automatically pinpoints the offending line of code and provides the function call context leading to it. In threading issues, where an error occurs in one part of the code but is triggered by another, the combination of the stack dump and a user comment reveals the nature, context and cause of the problem, which is otherwise very difficult to identify.

5.2.5. distribution

reViSiT is primarily distributed as a free download from the developer's website⁶. At times, the software has been included on magazine coverdiscs and CD shareware collections. More recently, it has appeared in major online shareware libraries, such as *CNET Download.com*, *Tucows*, *Brothersoft*, etc. To ensure their information remains accurate and current, a *PAD (Portable Application Description) file* is now maintained on site.

Two mailing lists, covering roughly 6,000 *reViSiT* users and all experiment registrants, are used to announce new releases. New versions are typically released at the beginning of the week, and only announced on the forum, so that initial use is largely restricted to experienced users and enthusiasts – those most eager to trial new features and best suited to catch teething problems. In the absence of any problems, a wider release is made the following Friday to catch people as they check emails before the weekend, when they are most likely to have time to use the program. The mailing list includes several press contacts and music technology websites, who subsequently carry the announcement to their readers, attracting new users and extending the surge of registrations and downloads.

5.3 system architecture

The user study begins with the collection of interaction data from the user's computer. The data is uploaded to a central server, from which it can be downloaded for collation and analysis. Figure 8 illustrates the overall architecture of the system and flow of data within the experiment.⁷

This section describes the methods and technologies used at each stage, detailing: the online procedure for registering new participants; the extensions made to the program under study (i.e. the *reViSiT* tracker) to record relevant data; the client- and server-side mechanisms for delivering that data to Cambridge. The tools used to collate, filter and analyse data from different sessions and users are then discussed in the next section. Although our focus concerns *reViSiT* and the tracker interface, the systems described here are designed to be easily-adaptable to other software applications and user environments.

⁶ <http://revisit.nashnet.co.uk>.

⁷ The code supporting the experiment is collectively referred to as “iMPULS”, an acronym for *Internet Music Program User Logging System*. The name reflects the experiment's attempts to capture the rapid interactions in the tracker user experience, as mirrored by the names of many tracker programs: *Scream-*, *Fast-* and, above all, *Impulse Tracker*, the original inspiration for *reViSiT*.

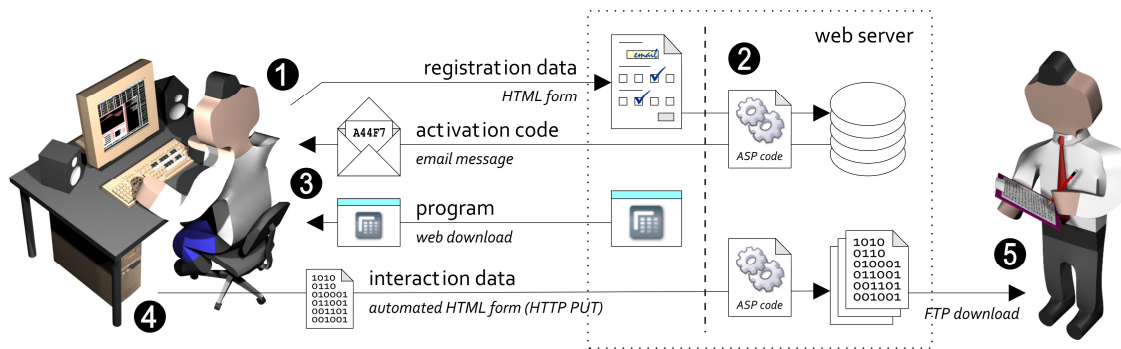


Figure 8 (1) The user registers by filling in an online survey and providing an email address. System architecture and data flow within the experiment (2) The server generates an activation code, using the email address and a unique ID. (3) The user downloads and runs the program, using the emailed code to activate it. (4) While the program runs, user interaction data is collected, then sent to the server. (5) Experimenters later download the interaction data for analysis.

5.3.1 user registration and identification

Participants register for the study through the experiment's website⁸ by filling in an online form and questionnaire (see Appendix C). Following submission, the server automatically sends an email containing an activation code to the address provided by the user. This code is used, by the user, to activate their copy of *reViSiT Pro*, and also, by the experiment, to uniquely identify the user. The user registration and identification process is summarised in Figure 8, illustrating how the user's identity is protected by separating the user's personal and experiment data.

reViSiT activation

To fulfil these roles, the activation code must:

- *uniquely identify a user without disclosing their identity*, to compare users and track their development, whilst ensuring an individual's privacy is protected;
- *discourage sharing of registration details*, to ensure data from each individual can be separated during analysis;
- *be computable instantly and automatically*, by the online server, to ensure a quick and easy registration process;
- *be verifiable without online access to a user database*, so the program can still be activated and used when an Internet connection is unavailable.

To achieve this, the code is a combination of two parts: a key derived from the user's personal identity, for their use in the program, and an ancillary ID representing an impersonal identity, for our use in the experiment. The user provides an email address to the website during registration, from which a cryptographic hash function generates the first part of the activation code. Hash functions are lossy, one-way mathematical operations, where the

⁸ <http://experiment.nashnet.co.uk>

input data is not recoverable from the result. As a result, the user then only has to provide their email address to the program for authentication against their code, after which only the code is used to tag experiment data, withholding the email address from the experimenters. The remaining part of the code contains a server-generated unique identifier to ensure there are no collisions as a result of the hash function.

Alternatives to requesting an email address were considered, since it could be construed as an attempt to gather personal information. For example, a network IP address could be easily and automatically retrieved from website visitors as they register. However, the rise in Internet users has led to the increasing use of network address translation (NAT), which effectively maps multiple users to single IP addresses. Another option would be to generate a hardware ID, based on the user's computer system. However, this is more complex and invasive, and tracks the system rather than the user, which breaks down if the user later modifies or switches the system they are using.

By comparison, using an email address holds several advantages: a user's email address is guaranteed to be unique; the information is easy for a user to recall and enter into the website or program; it needn't necessarily betray personal information; and is something many users are used to giving out. Additionally, it provides a means for the experimenters to collectively contact participants in the experiment, to notify them of important software or experiment developments. To support this, the mapping of IDs to email addresses are securely stored in a database on the server,⁹ which handles delivery of the experimenters' messages without divulging the email addresses or codes of individual themselves.

5.3.2 data collection

During use, the program records a variety of events relating to the user's interaction – both input and output. The data collected is as full and raw as possible, to support not only the planned analyses, but also allow for investigations that were not envisaged, without necessitating the further collection of data.

⁹ The storing of this mapping is also necessary for occasions when a user requires reminding of their activation code, since the server must remember which unique identifier was assigned to which participant. Theoretically, this enables a *brute force attack*, where an experimenter with access to the database could use it to generate all the possible keys from the email addresses and experiment IDs contained, and then compare them against the code used to tag specific interaction data. However, the separation of this database from the experiment data increases the effort required for such an attack, which can be simply averted by denying experimenters access to the database (e.g. having it maintained by a trusted third-party).

At the same time, recording all data is not feasible, as it would constitute too great an invasion of privacy and place a significant processing overhead on the client program. The collection mechanisms must not interfere with normal program operation or use, as might be the case if the experiment data required too much computer memory or processing power. Moreover, the data collected must be quickly transmittable over the Internet, as the program closes, without interrupting the user experience.

Table 1 – different event frequencies in the user experience

	in the order of...	frequency range
interaction	hertz (Hz)	up to 10 Hz
audio	kilohertz (kHz)	20 to 192,000 Hz
processor	megahertz (MHz) or gigahertz (GHz)	300,000,000 to 4000,000,000,000 Hz

In programming, the process of instrumentation is used to record and study program use, but can significantly reduce program performance, as timing data is collected and stored at such a high rate, for each executed function, line of code, or CPU instruction. As illustrated in Table 1, the frequencies of interaction are an order of magnitude lower than those relating to audio, which itself runs at a significantly lower rate, compared to the computer processor. As such, instrumenting a program to record user interaction need have little or no impact on the user experience, as long as it doesn't delay or interrupt other program processes, such as audio processing or disk access.

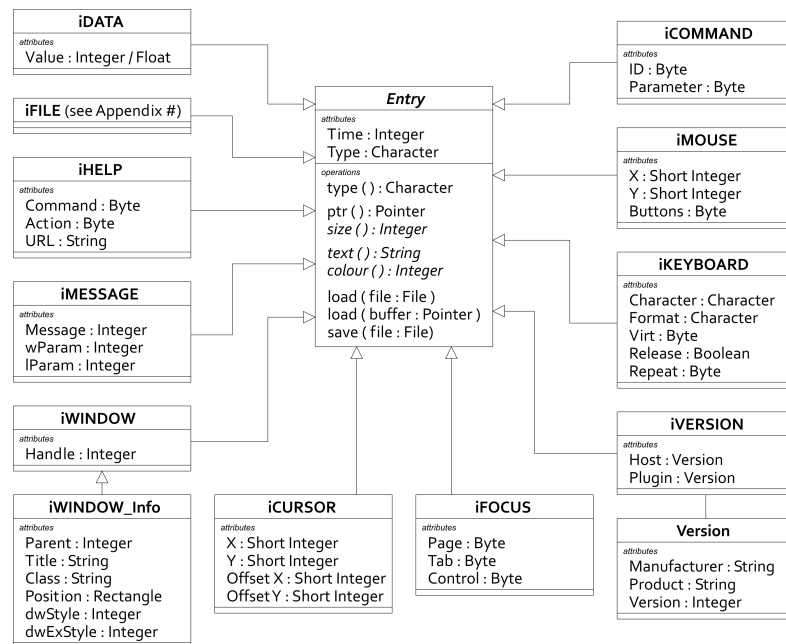
*data encoding
and bit-packing*

To ensure as small a processor and memory footprint as possible, events are *bit-packed* and stored in memory, then flushed to disk during periods when the computer is idle.¹⁰ Figure 9 gives an overview of the different log entries and data encodings used to record each type of interaction event. Further technical details of each event type are given in Appendix D.

The corresponding data types are derived from a base class, representing the members and functions generic to all interaction log entries. This abstract data type provides a single data member to identify the type of log entry, and declares pure and virtual functions that specify an interface, allowing code to handle

¹⁰ By default, the smallest data type (*datum size*) in the C++ programming language is 1 byte (8 bits), which is typically then aligned to 8-byte (64-bit) boundaries to improve the speed of memory accesses. In the best case, this means a simple true/false (1/0) boolean (`bool`) value takes 8 times the memory required (1-bit); in the worst case, it can take 64 times. To pack the bits more densely, *bit masks* and *Boolean operations* (logical AND, OR, NOT) are employed to address single bits within a byte (e.g. the value of the n^{th} bit of x is accessed using the expression $x \& 2^{n-1}$). At the same time, a dedicated *compiler directive* (`#pragma pack(push, 1)`) is used to override the alignment of members in the data structure used to record log entries. The remainder of the program is unaffected, and thus free to use faster, if more greedy, memory access methods.

Figure 9 – an overview of the different event types recorded as part of the experiment



collections of interaction events without worrying about the differing event types and their implementation or encoding. These functions require derived classes define code that:

- returns a human-readable description of the event (`text`)
- specifies a colour associated with the event type (`colour`)
- returns the object size (`_size`), for fast memory copying

Additional functions are declared and defined for the loading and saving of entries from file or memory, which can be overridden by child classes (for example, to save entries of variable length, such as those containing strings):

- loads event data from a file (`load(FILE*)`)
- loads event data from a memory buffer (`load(BYTE**)`)
- saves event data to a file (`load(FILE*)`)

instrumenting the user experience

The timestamp used for the session is set with creation of an `iMPULS` object, which hosts the functions, buffers and other mechanisms used to manage data collection.¹¹ However, hook functions and data collection are not started until the `iMPULS::start()` function is called, which should be triggered upon successful conclusion of the program's startup.

¹¹ The code to support data collection is contained in three files: a header file defining constants and parameters (e.g. connectivity settings) (`iMPULS_Constants.h`), a header file declaring data types and the support functions (`iMPULS.h`), and a source code file (`iMPULS.cpp`) providing the function bodies. The code is integrated into an existing program's source by including the main header file (`#include "iMPULS.h"`) and creating a single, global instance of the `iMPULS` controller object. These files and details about integrating `iMPULS` with other programs are available from the author upon request.

Hooked events (such as host notifications and help system calls) are recorded automatically, through callback functions provided by the `IMPULS` controller, but other events are recorded manually, using explicit calls to an appropriate `IMPULS` function:

- `keyboard(...)` and `mouse(...)`, called from the program's input handlers, upon user input.
- `message(...)`, called from the program's *window procedure*, upon certain Windows notification messages.
- `cursor(...)` and `focus(...)`, called as the user moves, within or between controls, tabs or pages.
- `command(...)`, called to log specific program functions as they are triggered (e.g. as the result of input), or activity not automatically caught by other handlers (e.g. occurring as a result of activity in the host, such as tempo changes).

Each function follows a similar procedure; constructing the log entry using the appropriate data type (see Figure 9 and Appendix D), then passing it to a function that adds the entry to the memory buffer, which is flushed to disk as appropriate.

5.3.3. data delivery

An interaction log file is created for each user session with the program. When the program closes, the file is compressed into a ZIP archive, which the program attempts to send back to the laboratory, over the Internet. In the event of failure, the archive remains on the user's computer until another attempt can be made, whenever the user next runs the program. Repeated attempts are not made immediately, to avoid interruption to the user experience or normal functioning of the computer. Should new log files be created before a previous one is sent, the file is simply added to the archive awaiting transmission.

offline uploader

Some computer musicians maintain a separate computer for their musical activities, separate from the Internet. In such cases, transmissions will always fail, log files will gradually build up, and the data may never be sent for analysis. Thus, when a specified threshold is reached, the program automatically compiles a separate upload executable, containing the collected logs, and prompts the user to run the package on a computer with Internet access. Until they do so, the prompt appears as a reminder at the beginning of each new program session, but can be dismissed after 5 seconds. When run, the new program simply uploads whatever logs it contains, before marking itself for deletion when the computer next starts.

To generate the upload program, the main program binary (i.e. `reViSiT.dll`) contains a template copy of the upload program as an *embedded resource*. Upon extraction, the new program contains an empty ZIP archive as one of its own resources, which the main program programmatically swaps for the archive containing the user's logs. The user is then prompted to save the resulting executable (.exe) to disk, and run it on any computer with an Internet connection (see Figure 10).

Ultimately, only 130mb of data was delivered via this method. Although this represents a small percentage (roughly 1%) of the total data collected, the threshold implicitly ensures that a minimum amount of data was gathered for each user, which increases the relative value of the contribution.

Figure 10
reViSiT Experiment
offline uploader tool



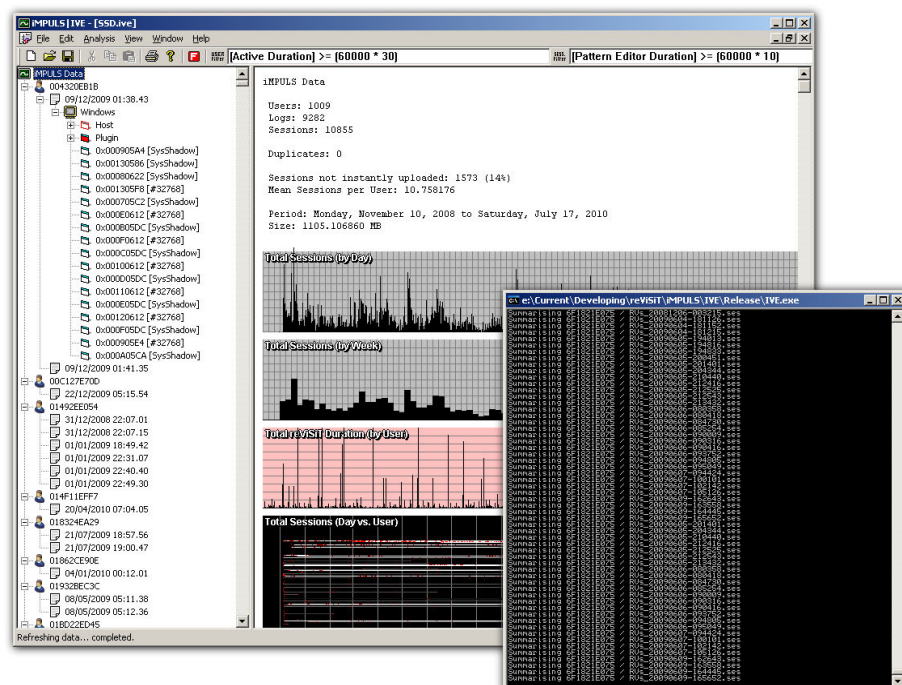
The development of *reViSiT*'s feedback and crash report system (see 5.2.4) allowed early testing of different methods of passing data over the Internet, avoiding interference from overzealous security processes. Normal file transfer protocols, such as FTP, are typically blocked; as is email (SMTP), whether using server authentication or not. In each case, default firewall configurations block the outgoing TCP port, requiring the user to manually open them, which we can't expect to happen.

The solution was to use basic HTTP data transfer (web communication), which firewalls don't block because it would disable most web access. The program therefore uses the built-in functions of the Windows Internet API to send files as though they were attachments on a web-based HTML form, generating the appropriate commands in the HTTP protocol (PUT, GET). The files are sent to an *Active Server Pages (ASP)* script, on a web server, where they are simply saved to the server's disk space.

5.4 interactive visualisation environment (IVE)

In order to efficiently manage the 20GB of collected data (see Appendix G for an overview), a dedicated program, *iMPULS|IVE* (*iMPULS Interactive Visualisation Environment*), was developed to download, verify, collate, filter, visualise the user logs, and support their analysis. The user interface is pictured in Figure 11, illustrating the main window, containing a tree overview (left) and object information (right), in which the data is presented and visualised. For operations that take time, a second window (inset) displays a text log, used to provide feedback during processing – reporting errors, showing debug info or the status and progress of analyses and other processes. Figure 12 presents an overview of the program's structure, which is further detailed below.

Figure 11
the *Interactive Visualisation Environment*

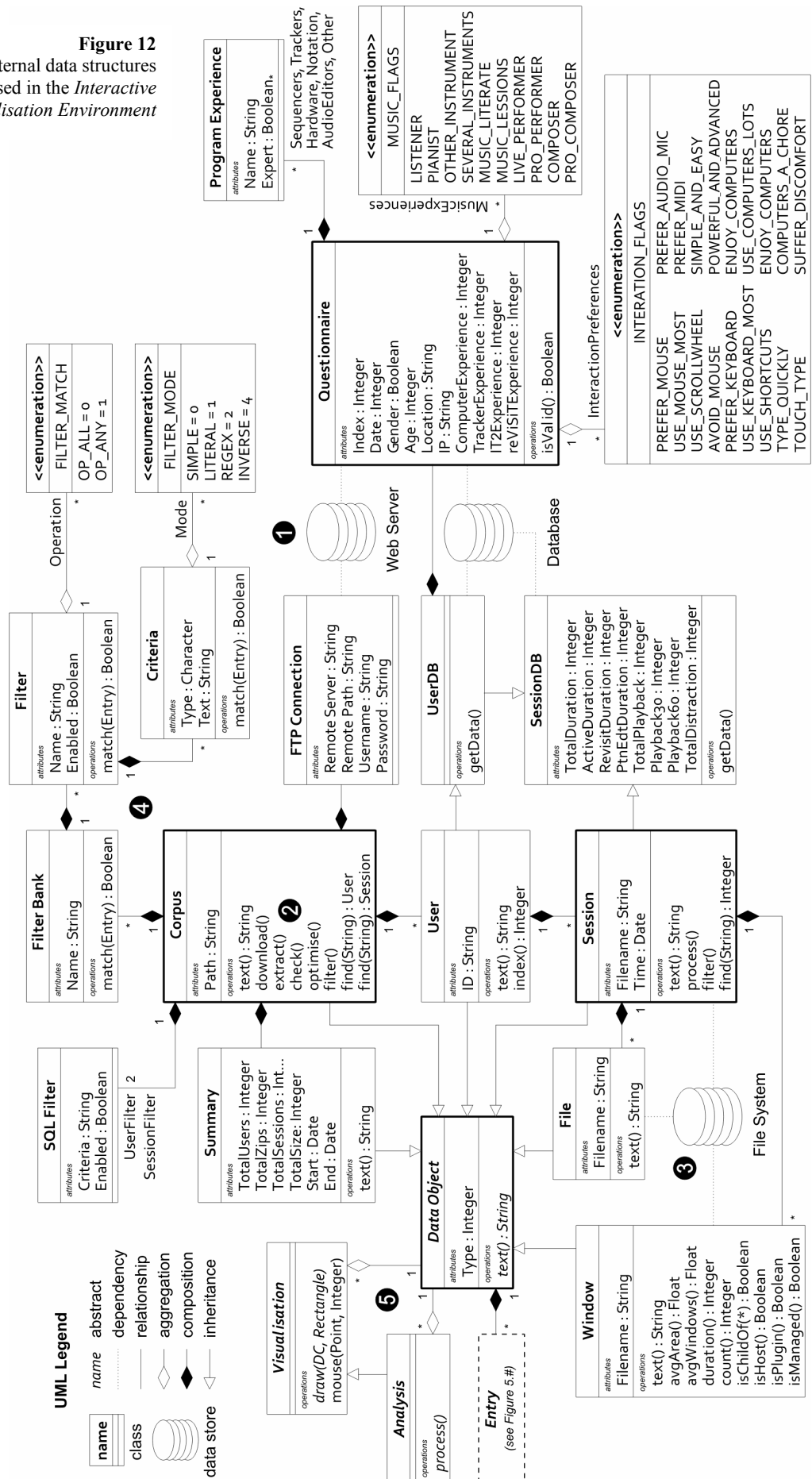


downloading and checking the logs

The program connects to the server containing the compressed logs, and downloads them to a local disk using FTP, grouping each user's logs in separate directories (❶ in Figure 12). Once downloaded, the logs are decompressed and then loaded to verify the integrity of the data (❷). The loading process reads events piecewise, checking each for encoding errors and corruption.¹²

¹² Unexpected values trigger an exception that rolls back the load process to the last known good event and tries to step over unrecognised data. If an error is detected or the loading of an event fails, the file pointer may not be aligned to the beginning of the next entry, preventing the loading process from continuing. Data corruption is rare, but can happen as a result of faulty hardware (RAM or hard drive) in the user's computer, transmission errors, or bugs in the encoding algorithms. Since a single misplaced bit can potentially invalidate hours of subsequent interaction data, the loading process attempts to recover from failures, iteratively trying to restart from offsets after the error. Invalid offsets quickly produce further errors, prompting the algorithm to move to the next.

Figure 12
internal data structures
used in the *Interactive*
Visualisation Environment



Upon successful verification, log files are processed into a new format, designed to accelerate loading and processing of the data, during subsequent analysis and visualisation. Larger entries, such as those detailing files (`iFILE`) or windows (`iWINDOW_Info`), are removed to separate files, and the remaining interaction events are saved to a *session (.ses) file* (③). While the program processes the events, it maintains a running status of the user's system – the current focus, cursor positions, modes and other activity (such as whether music is currently playing, or what the last triggered command was). A snapshot of this status is appended to each event before it is saved, so that the context of each interaction is known without having to search previous events. The resulting verified, and ultimately much smaller, session file can be completely copied into memory, enabling events to be loaded using very fast bit-copying functions (e.g. `memcpy`) and without further checks. Such optimisations significantly improve the loading times, allowing analyses and visualisations that are run over the entire dataset to execute in minutes rather than hours or days. This faster feedback cycle improves the provisionality of such activities, facilitating the exploration of different approaches.

5.4.1. visualisation and analysis

Although many analyses were already planned, the experiment was designed to support flexible exploration of the user experience – allowing different aspects of the interaction to be explored in greater detail, as their relative importance was established. As such, the tool was designed not only to manage the execution of the experiment, but to provide an experimental platform for testing new analyses and visualisations of the data.

Figure 13
visualisations in
iIMPULS|IVE
(see Appendix E
for specific details of
visualisations used)



Shneiderman’s Visual
Information-seeking
Mantra

visualisation and
the scientific method

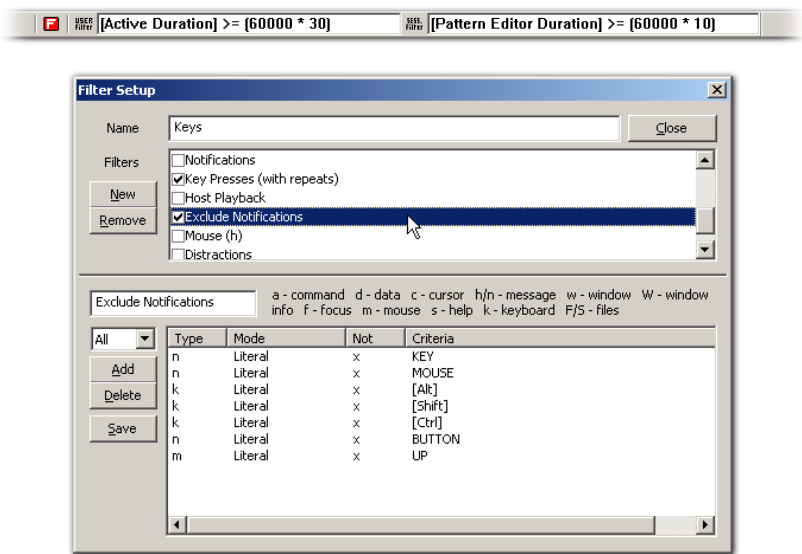
overview first...

In this way, the application provides capabilities consistent with Shneiderman’s *Visual Information-seeking Mantra* (1996): “Overview first, zoom and filter, then details-on-demand”. This exploits the visual-processing and pattern-matching capabilities of the human brain – providing as many different visual perspectives as possible and allowing the user to guide the visualisation process, in order to identify trends and relationships in data.

Visualisations, such as those in Figure 13, can suggest both new analyses and findings, but the added flexibility increases the risk of cherry-picking data – focusing (possibly inadvertently) on analyses that appear to support a specific conclusion or opinion, and overlooking those that produce less clear-cut results. When it becomes quicker and easier to perform analyses, it becomes easier to over-analyse data, tinkering with a methodology or sample until a finding is found. As a scientific tool, it is important to balance the use of visualisation in its capacity for exploration versus explanation (Tall, 1991). Appendix E details the visualisations used to support and guide analyses in subsequent chapters.

The main screen (Figure 11) presents the data in hierarchical (tree) form, with nodes for each *user*, containing nodes for each user’s *sessions*, which themselves contain additional nodes for *files* and *windows* described in the session. Selecting a node brings up information about the corresponding object in the right pane, which can also include summary information about the objects it contains. For example, the root node provides an overview of all users and sessions in the dataset; a user’s summary page presents details about the user and all their sessions. This hierarchy is explicit in both the interface and the internal data types used by the program (see Figure 12).

Figure 14
(top) user and session
filters on toolbar;
(bottom) interaction
event filter dialog



zoom and filter...

The tree hierarchy allows the experimenter to ‘zoom in’ on individual users or individual sessions, but other filtering systems allow them to restrict analyses or visualisations to groups of users or sessions (④ in Figure 12). Summary information for each user and session is cached in a database, and can be used to include or exclude users or sessions with certain properties. For example, Figure 14 excludes users with under 30 minutes total interaction, and sessions with less than 10 minutes time in the *pattern editor*.

Individual logs can also be filtered with regard to interaction events, limiting processing to specific event types (see Figure 14). The text representation required by the `Entry` class (Figure 9) enables events to be filtered using simple string comparisons – looking for combinations of key words or phrases that appear in, or are absent from, the description. Figure 14 also illustrates how different subsets of events can be extracted, combining several simple filters using logical operators.¹³

... then details
on demand

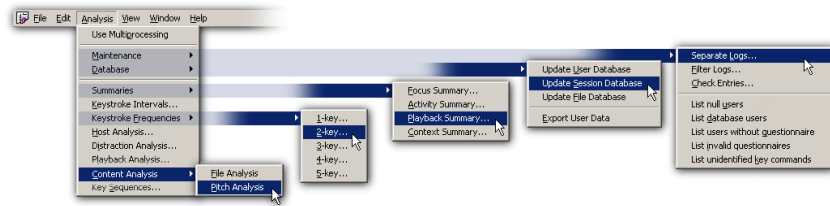
Once the dataset has been optimised and filtered, the program offers different ways to analyse the data (⑤), for visualisation or exporting to another program, such as *R* or *Excel*.

Analysis typically involves iterating over each user or session, extracting quantitative information about interaction events. For example, extracting the average keyboard input rate, for all sessions belonging to a user, exporting them to disk as tab-delimited or comma-separated values. Such interaction data can then be cross-tabulated with data from questionnaires (see Appendix C), enabling comparisons between users of different background and levels of experience. Alternatively, where a user has supplied enough data, similar observations can be made between their formative and more recent stages of development to look closer at the learning process – by, for example, looking at behaviour, averaged over set intervals – and the role of expertise.

As shown in Figure 12, analyses can be written for any *Data Object* type, and typically operate on the collection of entries they contain. As such, most analyses target the *Corpus* object, which contains all the data in the experiment – allowing access to all users and their sessions. An analysis is created by sub-classing the abstract *Analysis* class, and implementing the *process()* function. Compiler macros were written to abstract common or complex analysis operations, such as iteration or the use of multiprocessing.

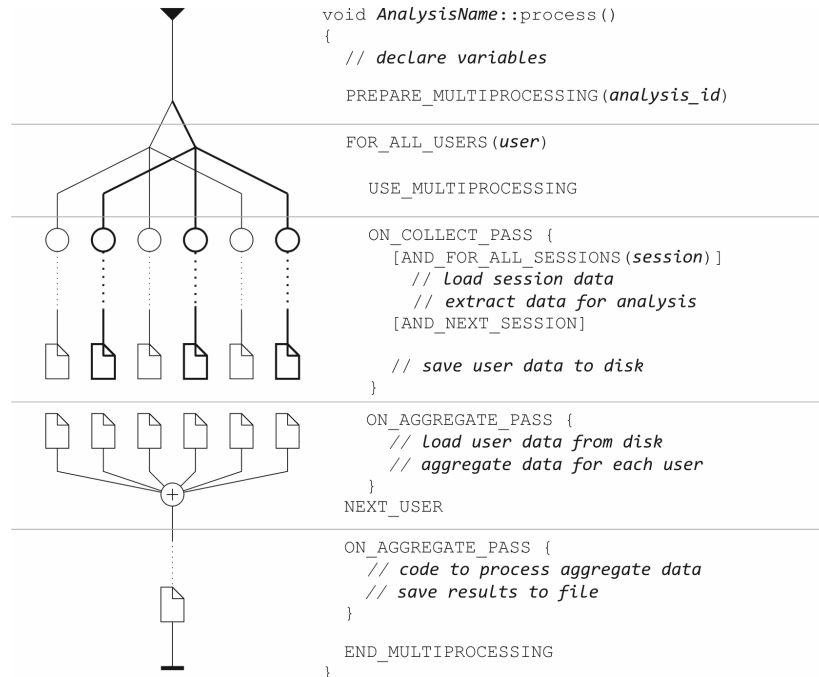
¹³ Such filtering helps researchers visually explore the data, but can also greatly speed up data analysis. For example, if an analysis only concerns keyboard input, the program can use the filtering system to extract `iKEYBOARD` events to separate files, which can then be analysed without loading the full session.

Figure 16 – analysis options in *iMPULS|IVE*



The user triggers analyses from the *Analysis* menu, shown in Figure 16. A wide range of analyses were developed for the *reViSiT* experiment, the specifics of which are detailed in the next chapter. Despite their diversity, most analyses follow a common procedure: loading, extracting, aggregating and exporting data. Figure 17 presents an example code template for a new analysis, which aggregates extracted data from sessions by user, and enables the use of multi-processing – allowing the computer to process more than one user at a time.

Figure 17
code template for data analysis, using macros (emphasis denotes separate process)

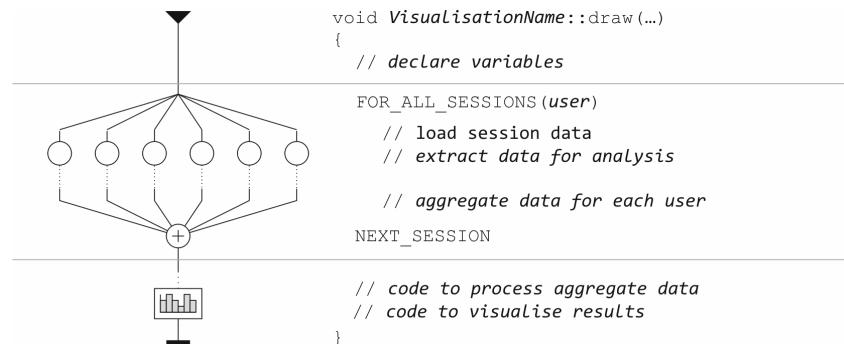


multi-processing

The multi-processing optimisation is achieved by performing the analysis in two passes – a first pass that collects data from separate users or sessions and saves it to a file, and a second pass that collects the data from these files and aggregates it. Since no session data is shared between users, the first stage, which includes the costly loading of session data, can be split between different threads. To implement this, the *iMPULS|IVE* program simply spawns other processes of itself, passing a command-line argument to them that defines an *affinity*. The affinity is an integer, defining which ordinals in a given collection (i.e. users or sessions) the process will handle. For example, in a two-process

scenario the original program creates one new process with an affinity of 1. With the original program assuming an affinity of 0, the two processes will thus divide the collection into the sets {0,2,4,6...} and {1,3,5,7...}, respectively. Processing is split using the `PREPARE_MULTIPROCESSING()` macro. Analysis is restricted to the appropriate ordinals, using the `USE_MULTIPROCESSING` macro, which can be placed inside either the user or session loops, to split processing by users or sessions, respectively. The compiler macros, `ON_COLLECT{...}` and `ON_AGGREGATE{...}` are then used to define what should happen in each of the two passes. The macros allow unnecessary technical details to be hidden from the experimenter, making it easier to follow the line of the analysis.

Figure 18
code template
for visualisation,
using macros



*preparing
visualisations*

The visualisations developed for the program (Figure 13) follow a similar template to analyses: loading, extracting and aggregating data – as illustrated by the example code in Figure 18. Instead of exporting the results to a file for use in another program, the code represents data visually, on screen. The program’s separate console window, inset in Figure 11, can also be used to quickly prototype visualisations, using low-fidelity ANSI text.

Visualisations are tightly integrated with the program, making it difficult to split the workload between separate processes. In many visualisations, the code itself still operates in two passes, where pre-processing is needed to establish drawing parameters – for example, in the case of normalising a graph where the maximum value must be known before the others can be scaled. While this makes processing slower, visualisations typically target single users or sessions, so there is less data to process – though more prolific users result in longer delays.

In addition to their use as an analysis tool, visualisations are an invaluable tool for monitoring the experiment and debugging the client program, as discussed in the next section.

5.5 running the experiment

This section briefly describes the final preparations, launch and running of the experiment. Table 2 provides an overview of major events in this process.


















2008	13 March	 <i>reViSiT 0.92.1</i> released to testers, with data delivery code.
	17 October	 Experiment development begins.
	15 November	 Experiment website launched.
		 <i>reViSiT 0.95 Pro</i> released to selected testers, with data collection code.
	1 December	 Experiment begins (announced on website / forum only).
		 <i>reViSiT 0.99.1 Pro</i> released to public.
	14 December	 Experiment announced to mailing lists.
		 <i>reViSiT 1.0 Pro</i> released, with full documentation.
2009	18 December	 Experiment announced in <i>Computer Music</i> magazine (Issue 134).
	6 January	 <i>iMPULS IVE</i> development begins.
	4 May	 <i>reViSiT 1.1 Pro</i> released, with user-definable keyboard shortcuts.
	23 May	 Over 1,000 experiment registrations.
	6 September	 <i>reViSiT 1.2 Pro</i> released, with high-definition pattern editing.
	20 December	 <i>reViSiT 1.3 Pro</i> released, with features for novices (e.g. mouse support).
2010	9 May	 <i>reViSiT 1.4 Pro</i> released, with sample and instrument library screens.
	10 July	 Data received from over 1,000 users.
	26 December	 End of Experiment Questionnaire issued (see Chapter 9).

Table 2 experiment milestones

5.5.1. testing experiment code

Testing of the data collection and delivery code ran in parallel with the testing of *reViSiT Professional* (see 5.2.4). In debug versions of the software, an additional console window is displayed, in which logged interaction events are printed as they happen, using *Entry*'s `text()` function. Log entries are created, encoded for saving, then instantly decoded for display, thus highlighting any problems in collection, encoding or decoding.

After basic internal testing, the experiment code was integrated with the *reViSiT Professional* versions already being tested by selected users. This allowed a wider variety of interaction events and styles to be tested, as well as a wider variety of user systems, with different Internet connections and security (e.g. firewall) configurations. It also broadened testing to include data delivery mechanisms, which had largely already been proven, through their use in delivering *reViSiT* user feedback (see 5.2.4). As more data was collected, work began on the *interactive visualisation environment (IVE)*; designed to analyse the data, but also enabling further verification and checking of the collected data (see 5.4).

5.5.2. experiment launch

The experiment went live on 1st December 2008. Like *reViSiT* releases, the launch was staggered, to minimise the impact of unforeseen teething problems, with the program, registration process, and wider experiment system. As such, the initial announcement was only made through the website and forum, offering *reViSiT 0.99.1 Pro* – tested and complete, but lacking documentation, which was added over the subsequent fortnight. On the 14th, the experiment and *reViSiT 1.0 Pro* was announced to the 6,000 users on the *reViSiT* mailing lists, by which time the majority of issues had been addressed.

In November, a press release was issued to several online and print music technology publications, to catch their January issues, due for release mid-December. The announcement was carried by a number of websites, and appeared in the News section of *Computer Music* magazine on 18th December, 2008. By the end of the month, over 500 individuals had registered.

5.5.2. maintaining the experiment

The launch was followed by an initial surge in registrations, as the novelty of the software and experiment attracted press coverage and people found time to try the software during the holiday season, and as existing *reViSiT* users migrated to the *reViSiT Pro*. As the novelty wore off, so did the number of registrations.

At the same time, only about a half of registrants went on to provide data – others either failing the activation process (e.g. providing bogus email addresses), overlooking the need for suitable host software, or using systems away from the Internet.

Furthermore, many registrants abandoned the program with only limited exposure. Although this was expected, the experiment objective was to observe users over time, as they developed skills with the program. Longitudinal information was assured from *reViSiT*'s existing users, but would only provide insight into previously-developed expertise. Thus, to increase the sample, it was necessary to keep the project, software and community active.

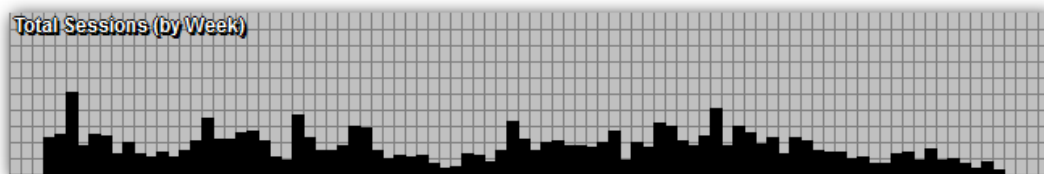


Figure 19 – user sessions uploaded, week-by-week

This was achieved through a series of updates to the *reViSiT Pro* program, addressing issues (including those exposed by the study) and adding functionality to broaden the appeal of the program, specifically to novices and new users. Each update prompted an announcement; restoring the experiment to the news cycle, increasing public exposure, and renewing interest. Appendix F details the updates, and the justification behind each. The overall success of this strategy is evidenced by the interaction spikes seen in Figure 19, each corresponding to new releases of the software.

Finally, in the closing weeks of the experiment's run, a second questionnaire was issued to gauge subjects' subjective experience of the experiment, as well as both sequencer (e.g. host) and tracker software in general, probing factors such as their experience of flow, use of notation, and changes in their interaction preferences or perceived level of skill. The questionnaire form is presented in Appendix C, results of which are detailed in Chapter 9.



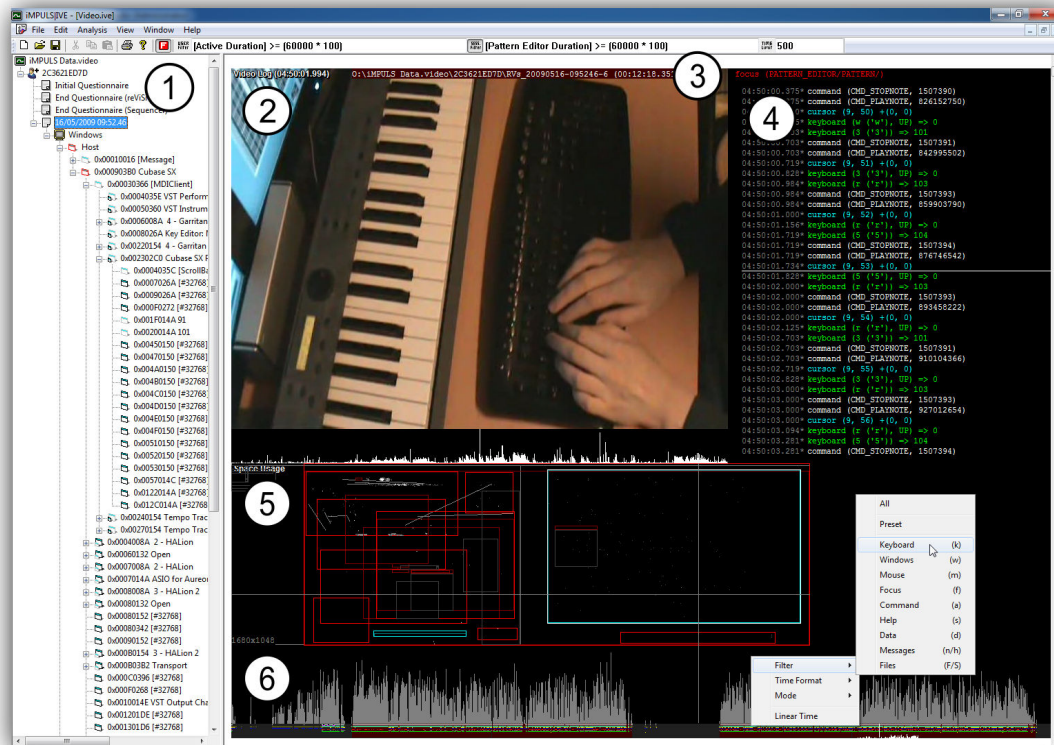
chapter six **video study: tracking composition practices**

As data collection proceeded online, a video study of an expert *reViSiT* composer was conducted to provide context for subsequent analyses. This section provides an overview of the interaction in the session, and makes general observations about the captured user experience, with regard to flow, virtuosity, and liveness (see 4.2.4).

In later chapters, these concepts are explored in more detail, using a much broader sample of users and scenarios. To this end, this qualitative, idiographic video case study not only provides a general overview of the tracker user experience, but formed an exploratory study that helped develop and focus later quantitative, nomothetic analyses, many of which seek to generalise findings made here.

*about the task
and subject*

A Dutch-based user who began using *reViSiT* in 2006 and since became involved in beta testing, the composer selected for the study uses the *reViSiT* tracker professionally, writing music for computer games, TV and film, but is also a well-known music artist from the *MSX* “demoscene” (see Section 2.2.2). Outside his professional work, for both enjoyment and practice, he also specialises in orchestral and FM synthesizer remixes and reversions of well-known electronic, film, and video game music. In this pursuit, the video records the composition of an original soundtrack for a *Warner Brothers* “*Road Runner*” cartoon, completed over the course of a single day (8 hours, with three 20-40 minute breaks), an intrinsically-motivated task the composer set himself. *reViSiT 1.3 Pro* (running in *Steinberg Cubase SX3*) was used and also provides the complete log of interaction during the session (see Section 5.3 for details).



- ① The **object hierarchy** shows users and sessions in the study, plus the window tree of the selected session. Selecting a window shows the map (see ⑤) of only that window and its descendants.
- ② The **video pane** displays and plays back recorded video footage, including audio. Videos are synchronised with both the window simulator and session log. Scrubbing is supported using the mouse scroll wheel, which can skip by events or fixed time intervals. Below the video, the audio waveform corresponding to the before and after the current frame is displayed.
- ③ The current **reViSiT focus**, the editor page and (where appropriate) tab or control.
- ④ The **session log** displays time-stamped interaction events immediately before and after the time shown in the video. Events are colour-coded by type (e.g. keyboard, mouse, focus change), and can be shown either as a sequential list, or spaced in proportion to their timing in the log.
- ⑤ The **window simulation** illustrates the changing configuration of the user's workspace, showing window positions and the current focus (in white), within the host (red) and reViSiT (blue) software, at 1:4 scale. Mouse usage is shown using points (for clicks) and lines (for drags), lightening the corresponding part of the representation. The simulation is synchronised with the video and log, flashing the appropriate window rectangle as it receives user input. See also Figure 4.
- ⑥ The **session overview** displays an overview of the entire session log, and can be used to move within the session. The lower strip shows the distribution of events within the session, colour-coded by type (as with ⑤). Sections with red background indicate accompanying video footage, which when active, also show a preview of the audio waveform. Right-clicking the strip opens a context menu with options controlling how the session log is displayed. Above the strip, a histogram shows the distribution of selected events within the session, based on the current event filter (see 5.4.1).

Figure 1 – Video analysis UI. Screenshot and description of the interface used to study interaction logs with video footage, within the *iMPULSIVE* program (see Section 5.4).

methodology

Following an initial review of the recording, interpretation of the video and log was supported by several discussions with the composer, which are quoted as appropriate. Figure 1 describes the *Video Analysis* screen of the *iMPULS|IVE* application (see section 5.4), showing a frame from the video recording. The camera is focused to capture the interaction around the hands and keyboard, where the majority of activity takes place.¹ The mouse is only partially visible at the top of the frame, but rarely used and largely restricted to rudimentary clicks in the host. However, all window activity and mouse input is captured using log data, and simulated visually beside the video. Corresponding events in the log, as well as histograms of selected interaction events, are also displayed.

comparisons with
other studies

It was hoped to conduct a similar study of sequencer use, but it proved difficult to locate a subject to provide a useful comparison: i.e. an intrinsically-motivated composer using the software to create and edit music, in contrast to professionals (working for an external goal and reward) or studio scenarios (which focus on hardware, rather than software, interaction). Instead, references are made to another longitudinal case study of a sequencer-based composer (Collins, 2005, 2007), enabling comparisons between sequencer and tracker approaches. A screen-captured video (with inset view of the keyboard) of the *Renoise* tracker, made by a *Renoise* user presenting a tutorial, is also referenced as appropriate.²

6.1 general observations

The vast majority of the user's time is spent at the keyboard: 98.8% of all tracker input is through the keyboard; even the host, which is little used, was typically manipulated through keyboard shortcuts (65.4% of host input), mostly for controlling song playback.

musical
touch-typing

Interaction is characterised by periods of sustained typing, with minimal body movements, punctuated by frequent auditions of short passages from the pattern currently under edit. Notably, the expert fluency shown in the video evokes common descriptions of tracking as a form of “musical touch-typing” (MacDonald, 2007). He moves around the pattern and music fluidly, maintaining a continuity of activity, facilitated by learnt actions, sequences and schema of screen and keyboard layouts, many of which are evident in the video. Like musicianship, these skills are developed through experience; they are easy to learn, but hard to articulate or teach, and are developed through repetition and practice.

¹ The narrower angle also served to reduce the subject's feeling of self-consciousness.

² <http://www.youtube.com/watch?v=SQ5jTaXywuM> [Last retrieved: 04/06/11]

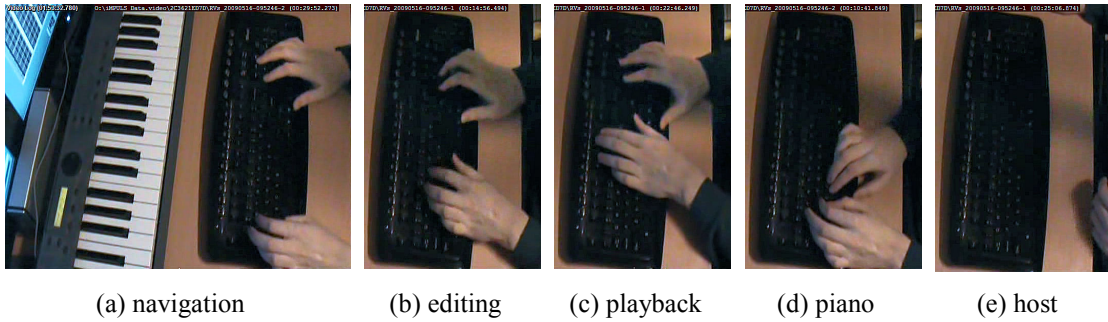


Figure 2
Common postures
observed in the video

- (a) **navigation** – default posture left hand over LShift/Tab (move between channels) and right hand over cursors (move between rows/columns).
- (b) **editing** – two-handed pose during most editing (left hand left of alphanumeric area, right hand over cursors);
- (c) **playback** – after editing, left hand moves to and hovers over F7/F8 keys, right stays with cursors (used to position cursor before and after, often Page Up/Dn). The division of hands allows editing, cursoring, and play-back to be dovetailed into one fluid motion, where one hand is in use whilst the other homes into position.
- (d) **piano** – both hands in alphanumeric area, for note entry, often overlaid or interwoven (as in piano fingering for chromatic runs).
- (e) **host** – withdrawal to the mouse for video-synchronised song playback and sampler settings (e.g. HALion). Playback triggered with left thumb on space, as left hand retreats from keyboard.

Figure 2 details five common postures adopted by the composer that illustrate his use of the keyboard. The *navigation* posture serves as the default “home” position between more active periods of pattern *editing* and *playback*, *piano*-like note entry and *host*-based song and video playback. The sound of fingers frequently brushing over keys is prominent in the video’s audio, suggesting the use of haptic (rather than visual) feedback, in guiding the hands around the keyboard.

Though these emerge as ‘set’ positions, the composer’s generic knowledge of program, keyboard commands and layout (spatial schemata) allow him to adapt to the context, to optimise the speed of editing, where the roles of hands are split between distinct roles. For example, during editing, or when the left hand is otherwise occupied, the right hand is used for control of playback.

At other times, though a hand will normally linger to terminate playback, listening can be mixed with other activities (navigation, editing). Here, playback and editing tasks are dovetailed, indicating a degree of parallel processing in the user’s thinking, and which is also evident in motor control.³ For example, where audio feedback prompts an edit, playback often continues; the editing activity monopolises both hands, and the termination of playback is

*playback and
editing threads*

³ Consequently, the sequential processing model of *KLM-GOMS* (Newell, 1990) is likely to overestimate the time taken for tasks by expert tracker users, which may be better modelled by the critical path method variation, *CPM-GOMS* (John and Gray, 1995).

deferred until a more convenient moment, whereupon playback is stopped with an almost inadvertent ‘stab’ at the F8 key. The F8 key is often also instinctively prefixed to the triggering of playback (e.g. F7), stopping and clearing the audio engine, whether such action is required or not.

Figure 3
Renoise in use
 Still from screen capture video, inset with view of keyboard and mouse (from online tutorial ²)



*bimanual
 tracker control*

While the postures in Figure 2 characterise the keyboard-centric design of the *reViSiT* tracker (as well as the earlier *Impulse* and *Scream Tracker*), a bimanual keyboard-mouse style is evident in other trackers. Figure 3 illustrates expert use of the *Renoise* tracker, originally based on the influential and more graphical user interface of *Fast Tracker*. The mouse and pointer, rather than the keyboard, are used to navigate around the program and data (pattern). However, the division of responsibilities between the hands is consistent with that seen in *reViSiT*. In the video, the user spends more time in visual search and is frequently forced to home the right hand between the mouse and keyboard, for navigation and editing tasks respectively; but the mouse integrates well with the tiled and graphical elements of the UI around the pattern editor, which accommodate greater and more varied functionality than that found in *reViSiT* – also meshing well with the generally mouse-based interaction with plugins.

*minimal
 window use*

Figure 4 illustrates the use of windows within the host workspace, across the composer’s multi-display system. The use of two high-resolution (1690x1050) monitors minimises the contention for screen space, allowing him to dedicate the entire right screen to the *reViSiT* window, using the left as a peripheral display for the host’s windows, such as the project / arrange window, tempo track, synthesizer settings (*HALion* sampler), mixer and video preview.⁴ As such, the physical desktop layout, with the keyboard before the

⁴ The unused area, left of the desktop, corresponds to a space reserved for an IRC chat client, used to maintain contact with other members of the MSX demoscene, but largely ignored during composition.

minimal window
management

avoidance
of the mouse

right monitor, affords the composer the impression of dedicated, keyboard-controlled tracker system, and serves to contain his attention and focus.⁵

In his rare use of the host, window focus and mouse interaction is concentrated in the project window, manipulating song position (for playback), configuring tracks and sound sources, and infrequent track editing. Aside from positioning feedback displays at the extremes of the workspace, little consideration is given to maintaining optimal layouts or maximising use of screen space, where hidden or overlapped windows are instead simply brought to the foreground as needed, from a cascade of windows in the upper-left quadrant of the frame. Notably, though most attention is given to the project window, it extends across less than one third of the application, which little effort is expended to redress.

Mouse use in *reViSiT* is similarly rare, and simply appears to serve for returning the window (and keyboard) focus after excursions to the host. The window isn't maximised, making room for the sequencer's transport bar, which is only used in a feedback capacity, to provide information about, rather than control of, song and video playback. Nonetheless, a notable amount of screen real estate around the *reViSiT* window remains unused during the several hours of interaction, which could easily be reclaimed by sizing the window. Along with the host, this highlights the composer's antipathy towards mouse use and the management of floating windows, which was also evident in discussions.

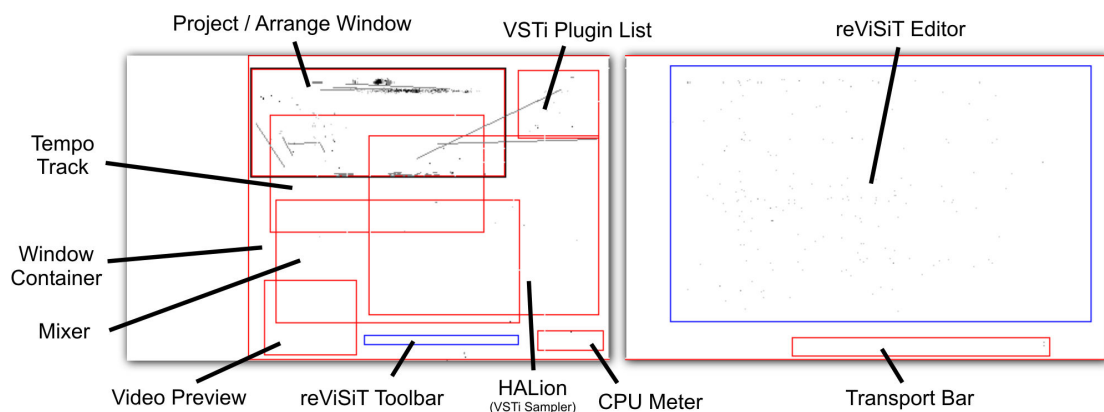


Figure 4
Window simulation

Windows are illustrated as rectangles (red = host; blue = reViSiT). Mouse activity (clicks = dots; drags = lines) is shown by a monochrome white-black gradient relative to each window, where each pixel in the simulation represents a 4x4 pixel area on the user's screen.

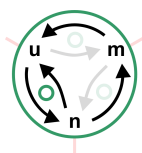
⁵ *Cubase's* transport bar also resides on the right monitor, beside *reViSiT*, but is used exclusively for visual feedback during song and video playback, rather than control of it.

Interaction begins with an initial period of playback and preparation, as the composer familiarises himself with the video, aligns patterns to events in the cartoon, and loads samples in the *HALion* sampler. This period is characterised by mouse interaction in the host. Once complete, mouse use is largely limited to repositioning the host song pointer before playback, and otherwise avoided by the composer, who notes that “mouse usage for creative things is a problem” and only accepts its role in the sampler because he doesn’t try to use it creatively.

After this point, the composer spends almost all his time with both hands on the keyboard. Despite the MIDI keyboard beside it, the computer keyboard is used for pitch-entry. Similarly, though his studio contains a control surface, mixer, and many other MIDI synthesizers and keyboards, they remain unused.⁶

After preparation, just over 15 minutes are spent recreating the *Warner Brothers* theme tune for the start of the cartoon. Largely an exercise in musical transcription, this period is characterised by higher interaction rates and productivity, quickly producing a fully-orchestrated arrangement of the jingle. During this period, the composer does not reference an original recording or score of the music. Instead, the composer enters and edits the music using audio feedback to build a copy of the piece from memory, experimenting with edits and identifying mistakes by ear.

composing by ear



“expand/explore”
approach

Consequently, audio feedback is in constant, frequent use, during interaction. Playback commands follow even small edits, where it is clear the composer uses the audio to understand the music he has written, relying less on the visual notation. This illustrates the central role of audition in **manipulation-driven** notation systems (inset, see Figure 4-9). Occasionally, there is more sustained editing between auditions, when the sound, he says, is more predictable.

This intuitive, exploratory approach to composition is evident throughout the session, as the composer works linearly, drafting and finishing small, sequential sections, rather than creating a blueprint for the whole soundtrack. In subsequent discussions, he observed that the practice of working in small sections is common in tracker users, in contrast to sequencer users, who tend to build pieces in layers (e.g. tracks), commenting:

Actually, unless I'm remixing or rearranging an existing piece, I'm never planning ahead. I don't plan large things. I expand/explore small things.

⁶ Indeed, the composer notes that they have not been used in months.

FOCUS & ACTIVITY

Host (Cubase SX)
 ■ % time in focus
 ■ (music playing)
 — input (avg. cmds/min)
 (measured from top)

reViSiT
 ■ % pattern editor
 ■ (music playing)
 ■ % other screens
 ■ (music playing)
 — input (avg. cmds/min)

USER INPUT (cmds/min)

Cubase
 ■ keyboard
 ■ mouse

reViSiT
 ■ keyboard
 ■ mouse

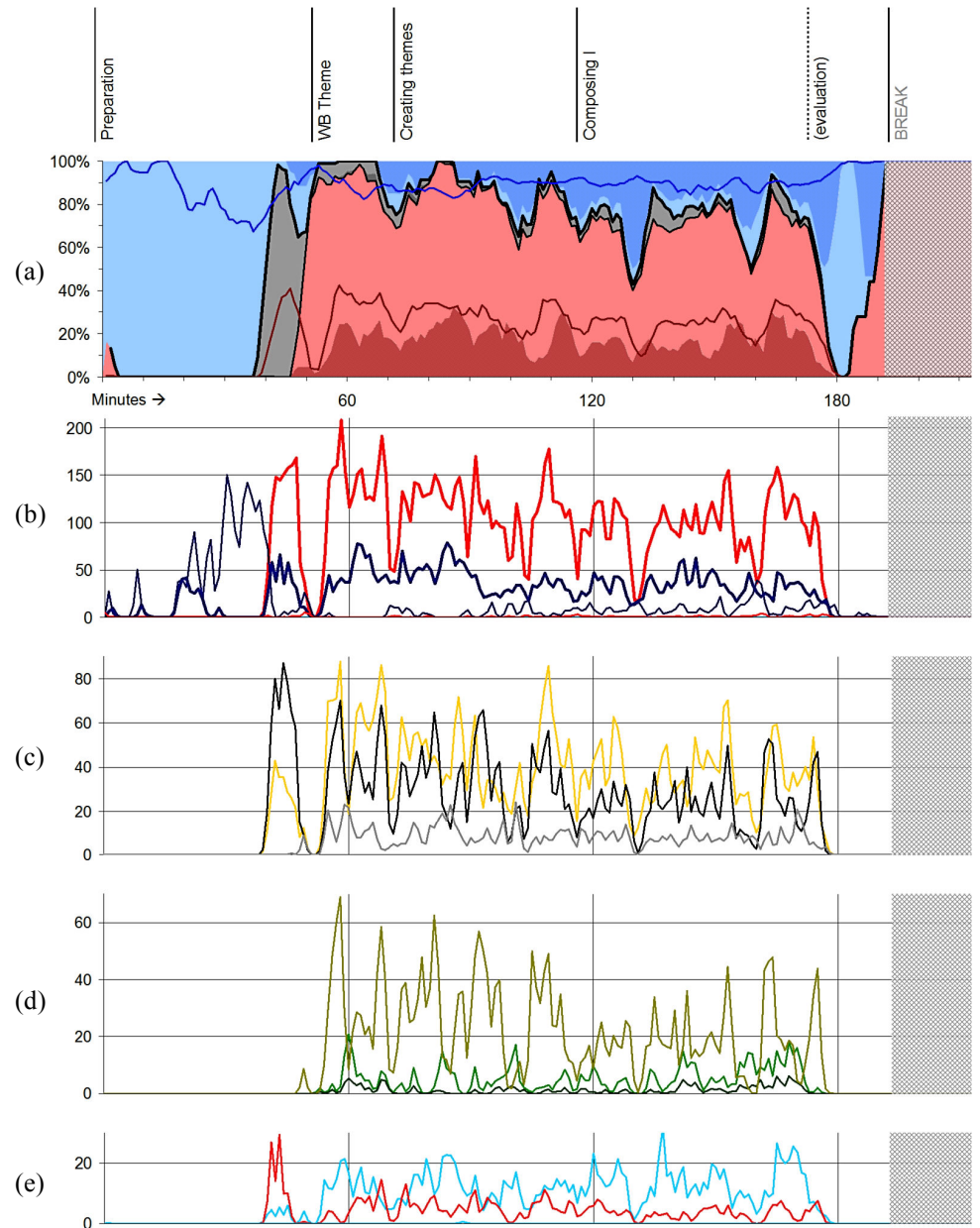
reViSiT INPUT (cmds/min)

Based on contexts of key commands in reViSiT use (see 7.3 and Figure 7-5).

— EDIT
 — NAVIGATION
 — PLAYBACK

— DATA
 — SELECTION
 — CLIPBOARD

— FOCUS
 — SETTING



In reference to *vertical and horizontal composition* styles (Folkestad, 1996), which respectively correspond to initial focuses on harmony or melody, this approach brings both considerations forward in the composition process. Within the tracker's pattern architecture, the composer is seen to work horizontally, laying down short excerpts of melody, then augmenting it with harmony and even final touches, before moving to the next pattern. Thus a whole song becomes the product of many smaller, sequential creative processes, where each pattern goes through Sloboda's progression from an initial draft form to final score (Sloboda, 1985). Moreover, new themes are rarely auditioned away from the pattern (e.g. in the *Instrument List*), but entered and experimented with in-place, in the pattern itself.

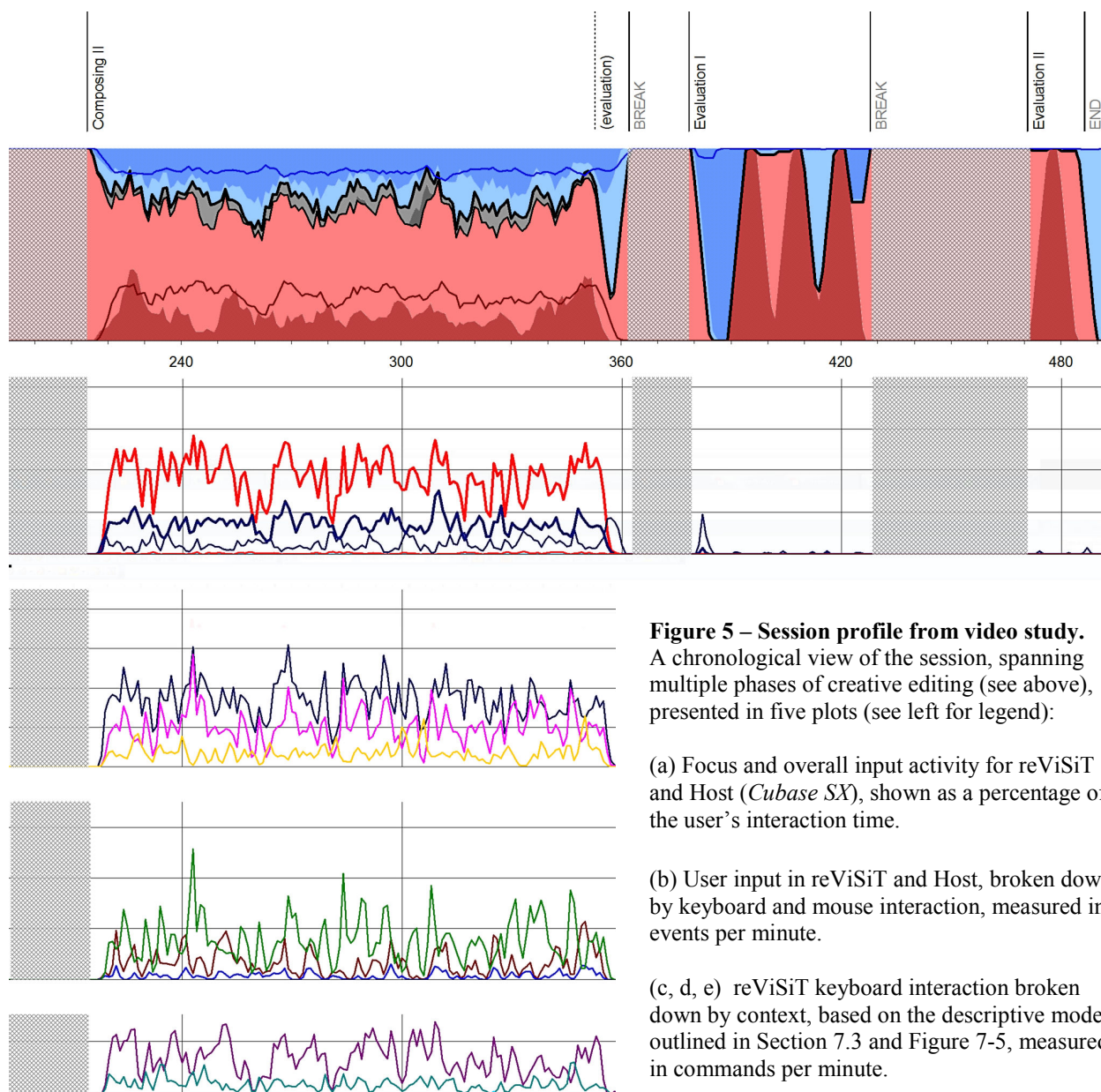


Figure 5 – Session profile from video study.

A chronological view of the session, spanning multiple phases of creative editing (see above), presented in five plots (see left for legend):

(a) Focus and overall input activity for reViSiT and Host (*Cubase SX*), shown as a percentage of the user's interaction time.

(b) User input in reViSiT and Host, broken down by keyboard and mouse interaction, measured in events per minute.

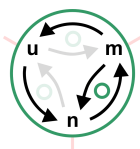
(c, d, e) reViSiT keyboard interaction broken down by context, based on the descriptive model outlined in Section 7.3 and Figure 7-5, measured in commands per minute.

linear working style

While constructing patterns could be approached using trial-and-improvement, the composer's ability to maintain a coherent musical thread between patterns, and across the piece, demonstrates a deeper musical understanding. Few mistakes or major corrections are evident; the composer enters the majority of a section in sequence, and seems to have clear idea of what he wants, and how to realise it (an example of *Clear Goals*, in flow; see Section 3.7).

Though the user has had considerable exposure to music performance (including piano tuition), his composition practice is largely self-taught; implicitly learnt over many years of working with trackers ("no training; just looking, listening, seeing and understanding the relation"). This has lead to an extensive, but tacit knowledge of musical processes, which he finds difficult to

“macro-listening”



“spot-on debugging”

near-realtime
composition

articulate. For example, asked how he knows patterns will work together, he states, “If the expanding/exploring is done in a natural way, then it’ll fit”, and says that he simply relies on listening to check that a section feels “natural”.

To gain a broader perspective of the music, the composer devotes long periods to repeated playback of the wider song (often more than 30 minutes in length), which he calls “macro-listening”, contrasting to the shorter auditions supporting editing (“micro-listening”). During this time, the task switches from composition to active listening (a *realisation-driven* system; inset, see Figure 4-9).

He also makes extensive use of selections and clipboard, allowing him to work with larger blocks of music and repeat elements of the music, to form progressions. However, users with less experience, lacking such knowledge and technique, may find it harder to maintain themes and ideas across the breaks between patterns.

Moreover, this linear workflow may be a consequence of relying on audio, rather than visual, feedback – where the poor *role expressiveness* of the text notation makes it harder to step back and quickly gain a broad overview; with audio, longer perspectives entail longer interruptions, as the song plays in realtime.

Instead, the composer uses short excerpts of playback to guide edits, and only listens to it in its entirety towards the completion of a phrase. Here, listening becomes the focus of interaction, as he triggers playback (F7) with his left hand and leaves it poised over the adjacent stop key (F8), ready to terminate playback and jump straight into editing, cursoring with his right hand, as soon as a mistake or new idea becomes apparent. The composer calls this technique “spot-on debugging” (in reference to similar approaches in programming, such as *just-in-time (JIT) debugging*), a further example of the primary role of musical feedback in the tracker.

Fast navigation around the music and program is central to the composer’s working style. Rapid, complex cursor activity, seamlessly interwoven with almost every task (including listening, note entry, arranging, and instrumentation) frequently exceeds rates over 100 cmds/min. In spot-on debugging, for example, cursors are used to quickly select the playback material and then to quickly convey the composer to the appropriate point, when he hears something.

During note entry, the cursor is also used to step through the pattern to correctly place notes. Unlike a live recording, notes are not entered in realtime, but the composer’s dexterity in interleaving cursor movement with note entry allows him to preserve much of the rhythm of the notes, so the character of the melody or phrase is preserved in the incidental audio feedback. At the same time, the

*arrangement
and abstraction
in clipboard use*

lack of rigid metre allows him to slow down or pause as necessary, for more complex edits. Faster-than-realtime input is also possible, and it is not uncommon to see longer passages initially entered into a confined space, then expanded using shortcut keys.

In the session profiles (Figure 5), some editing periods are characterised by direct data entry, and others by increased use of selections and the clipboard. Frequently, the composer is seen to edit a short section in detail (a beat or bar) before cloning it to form the basis for longer phrases. Though this practice is common in loop-based music, leading to progressive musical styles⁷, the composer uses this approach for more intricate musical structures, whereby the flexibility of block selection and the clipboard allow him to build new patterns not just by repeating whole sections, but by drawing on and mixing select parts of previous material, in a process more like *bricolage* (see Turkle and Papert, 1992). Unlike individual notes, selection-based edits do not automatically trigger audio feedback, so the composer relies more heavily on short excerpts of song playback and “spot-on debugging”.

After the composer has laid down several basic themes, a slight shift towards increased clipboard use occurs (~02:20), continuing until the end of the session, as new material increasingly draws on that preceding it. During selection use, the interaction rate remains high, and with each key command now affecting multiple notes, overall productivity increases. Block selection supports a subtly higher level of music editing that mixes microscopic note-level editing with more abstract editing based on themes, phrases, parts, and other musical devices. This transition is implicit, with little change in interaction style (input mode, visual representation), thus enabling free movement between stages in the creative process. So, even as a user moves from exploratory creativity (finding themes) into a later-stage composition process based more on problem-solving (arrangement, applying music to video), there is little to hinder them from experimenting with new ideas.

*host-based song
& video playback*

Working with video, the composer is forced to return to the host program to audition the song in-sync with the visual footage. This diversion punctuates longer periods of interaction with the tracker, in which the music is created and edited. As a result, a clear distinction in the role of playback emerges between the host and *reViSiT*, whereby the sequencer provides the longer, broader musical context, managed through the timeline and transport bar, and the tracker provides focused feedback for editing, through the

⁷ Music based on a progression, where several iterations of a passage are gradually developed or varied, in respect of melody, harmony, rhythm or texture; common in dance, house, trance, drum’n’bass music.

keyboard. These two modes of playback differ in frequency, duration, and manner of control, as well as the subsequent posture of the user. In the tracker, the composer continues to interact or hovers, poised over the stop key, in anticipation of further editing. In the sequencer, the composer positions the playback cursor with the mouse, and triggers the song with the keyboard – striking the space bar with his left hand, as it retreats from the keyboard – and then remains idle, listening to the music. In this scenario, the sequencer’s role is that of a tool for evaluation, the final stage in the creative process. Later analyses explore this in the context of what other studies (Blackwell and Green, 2000; Smith *et al*, 2009) have identified as a tendency for music software to focus on the later stages of creativity (i.e. transcription, productivity).

*energy and
tiredness*

At the same time, longer auditions can be restorative. The composer noted that the rapid interaction and constant focused attention of tracking can be tiring, disposing him towards longer auditions as a productive means of resting. The intense, hard cut bursts of sound arising from frequent auditions of notes, passages and patterns may also lead to ear fatigue, though longer breaks after several hours of interaction help to combat the risk.

*centralised
focus & control*

Within *reViSiT*, the composer spends the vast majority (93.8%) of his time in the Pattern Editor. Apart from the initial configuring of instruments and occasional edit to the Pattern Order, the only significant use of any other part of the program is the Instrument List’s role in changing the current instrument used for editing.⁸ In *reViSiT*, there are a number of ways to do this from the Pattern Editor itself, and while the Instrument List may have its advantages, the composer concedes that his choice of method is likely a habit picked up in IT2, from which the original inspiration for *reViSiT*’s UI comes. This is a clear indication of well-learnt interaction, based in the development of both motor skills (key sequences) and spatial schemata (the instrument list and keyboard layout).

*mastering
the tracker*

The composer is conscious of his expertise; as something that has taken years to develop and mature, largely learnt through practice and experimentation, but also through dissecting the music of others’ and the sharing of tips and tricks in online communities (e.g. the demoscene). When asked to reflect on the most important concepts and lessons a new user should learn to develop mastery of the tracker, in comparison to other digital music practices, he cites (in no specific order):

⁸ More usually, he "picks up" an instrument from existing music in the pattern, by moving to one of its notes and hitting Enter. This way, the visual search through Instrument List is avoided, and the user’s attention can remain with the editing context.

- *the freedom and blank canvas of the pattern*
to place any note(s) of any instrument in any cell or channel, allowing the composer to group elements as they see fit, without being bound to or separated by MIDI channels, or having to create and prepare tracks before data can be input;
- “*spot-on debugging*”
the rapid edit-audition cycle and use of editing cursors to quickly trigger playback (F7), during which the user listens and remains poised, ready to jump back to editing;
- *fast navigation using the keyboard*
allowing routes through the program, commands, and sequences of actions to be executed from memory without visual inspection, and fluidly interwoven with other inherently keyboard-based tasks, such as editing.⁹

6.3 evidence for flow and virtuosity

Evidence of several flow components (defined in Section 3.7) emerge from the video, log data, and discussions. The linear approach to composing music in patterns sequentially – without significant backtracking, and as opposed to establishing and building on an outline – demonstrates the existence of *clear goals*, which the composer knows how to achieve using the program, confident in the *balance of challenge and ability*.

*focus and
feedback*

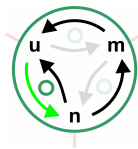
A high rate of interaction is sustained over several hours, maintaining *concentration and focus*. Specific techniques, like “spot-on debugging”, help keep the user engaged and absorbed in the editing process, providing *direct & immediate feedback*. Through similar expert use of the keyboard, he is likewise able to maintain a strong *sense of control* throughout.

*action-awareness
merging*

On viewing 5 hours of footage from a single working session, he was surprised not only by the length of time he had been working, but to see how “obsessed” he was, comparing his typing to “speedcubing” (competitive *Rubik’s Cube* solving). This extreme level of engagement indicates *action-awareness merging*, from which he exhibits a *distorted sense of time* and *lack of self-consciousness*, commenting,

*I’m never conscious of those kazillions of keyclicks [...] It’s also as if it’s very long/boring. I was almost afraid that this vid’ showed tracking is *not* fast, but alas, when in the first 18 minutes I have a full orchestra/bigband ... I guess it’s still radically fast.*

⁹ The composer cited an earlier occasion, in the MSX program *FAC SoundTracker*, where his knowledge of the program and reliance on audio feedback enabled him to continue using the program several days after his monitor had stopped working.



The barrage of sound in these editing sessions may seem discordant to observers, as the disjointed playback jumps randomly and fleetingly between short excerpts of the music. However, the subject remains unfazed, again indicating his *concentration and focus* and *loss of self-consciousness*.

Perhaps most importantly, the fact that the subject voluntarily spends 5 hours of tiring, engaged interaction on a musical exercise with no promise of *extrinsic reward*, seems to point to an inherently enjoyable, *intrinsically-rewarding* flow experience.

From observations, interaction data, and subsequent discussions, it is evident that the composer is able to use the tracker as part of what he sees as an intuitive (“natural”) approach to composing, where his focus and expertise enable him to quickly sketch and refine (“explore-expand”) musical ideas in notation, guided by the frequent and integral use of audio feedback. The tracker, through its use of the keyboard, enables the development of motor skills that enable rapid and fluent interaction bridging note entry and music editing with program control. In this example of constructive **flow interference** (inset left, explained in Section 4.3, Figure 4-9), focused interaction with the notation is supported by both visual and musical feedback (*flow redundancy*), though manual skill is required to fluidly integrate them in the user experience.

In the logs, these skills and working styles are manifest in several ways, such as the rates of interaction, fluidity of input sequences, as well as frequency and use of musical feedback. In the following chapters, these quantities are among those explored using logs and feedback from other users of the *reViSiT* program, in an effort to build a broader understanding of flow and virtuosity in general use of music software. Specifically, Chapter 7 looks at the users’ development of motor skills with the computer keyboard, notably through which a rapid edit-audition cycle becomes possible. This skilled use of musical feedback is further detailed in Chapter 8, which explores how a greater frequency of feedback contributes to greater liveness in the user experience (see Section 4.2.4). Chapter 8 likewise explores the role of visual feedback, and factors that affect a user’s focus and concentration. Further components of flow (see Section 3.7) are examined in Chapter 9, which combines earlier findings from the video study and user logs with additional survey results, working towards a more general model of how a program’s capacity for flow is determined by specific properties of the notation.



chapter seven **keyboard use and motor learning in tracking**

The use of the keyboard is central to tracking, distinguishing it from the more common mouse-based GUIs used by sequencers, DAWs and score editors. The keyboard's distributed, fixed layout supports motor learning that enables rapid rates of interaction, and control over a broad range of program functionality. In many trackers, all tasks are executable through the keyboard, including note entry and editing, block selection and clipboard arranging, playback and program management.

This section looks at several aspects of keyboard interaction, across varying levels of experience. Following a simple look at speed and the rate of interaction, other aspects of timing, such as rhythm, are explored. Performance metrics are then integrated with accounts of keyboard and program knowledge, such as command vocabulary and fluency, using a descriptive model of tracker interaction that generalises tasks in music software, to illustrate the development of technique in the tracker.

The findings and methods presented in this section should be generalisable to other music hardware built on similar styles of interaction, such as MIDI controllers, instruments, and control surfaces with multiple, fixed-function controls, plus other *space-multiplexing* input devices, as opposed to *time-multiplexing* devices, like the mouse (Buxton and Myers, 1986).

7.1 speed and timing

The average user demonstrated a keyboard interaction rate of 9.74 ± 0.44 commands per min (cmd/min). Tracker novices were the slowest, averaging 6.34 ± 0.68 cmd/min ($n=67$), and tracker experts were significantly faster ($p < .05$), averaging 11.89 ± 0.50 cmd/min ($n=107$) – almost twice as fast as novices. However, the fastest overall work rate is demonstrated by reViSiT experts, who can average up to 42.42 ± 1.08 cmd/min (exhibited by the composer who took part in the video study).

These figures average the total number of keyboard commands triggered over a normal period of reViSiT interaction, which also includes thinking time and periods spent interacting with the mouse. Sessions of over 30 minutes are used to calculate a user's average, ignoring the first 10 minutes, which is characterised by preparatory activity. In most users' first session, bursts of data entry are also common in the first 2 or 3 minutes. This is attributed to new users entering random notes into the pattern, to experiment with the workings of the pattern editor and keyboard – similar to when users record random music into a sequencer, to test its workings. In both cases, the provisionality of the notation enables the user to learn by experimentation.

Figure 1 shows log graph (with linear detail inset), showing the timing separation of different keys in sequences of keyboard input (within a 10s threshold, and ignoring repeats)¹, as a measure of the speed users move around the keyboard.

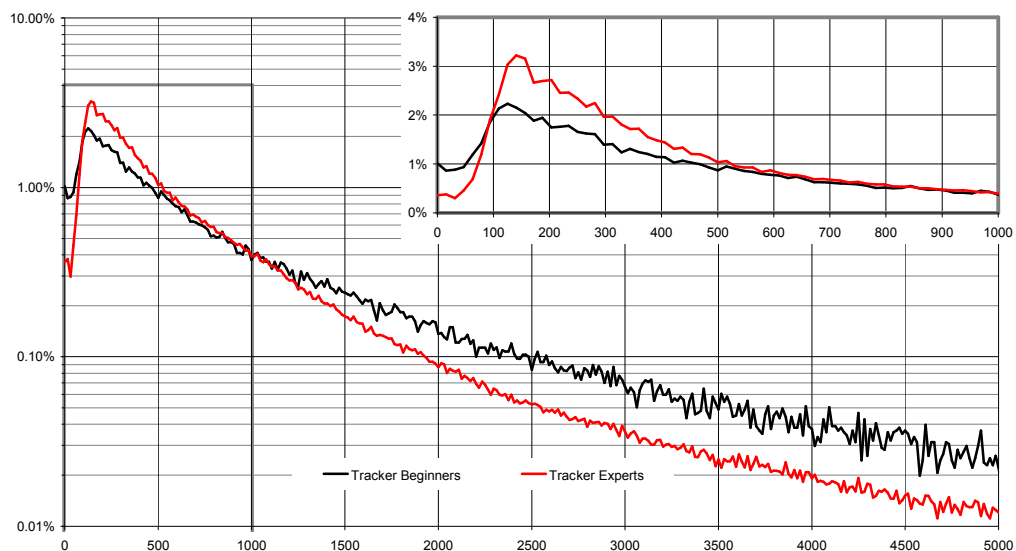


Figure 1 – Distribution of intervals between distinct keys (ignoring repeated keys).

¹ Separations greater than 10s are taken to indicate a pause or break in interaction.

*experience improves
speed and consistency*

Both series decay according to an inverse power law (beginners, $R^2=0.963$; experts, $R^2=0.987$), but while experts average a faster overall rate of interaction (median = 400.9ms, compared to 557.4ms for beginners), the mode drops 11% (from 125.0ms, for beginners, to 140ms, for experts). Instead, experts' higher average is attributable to an increase across the 100-500ms range and decrease in longer intervals (above 1000ms). Two explanations are offered for this: firstly, that the higher median rate for experts leads more quickly to tiredness and a long-term slowdown in performance. Secondly, that experts do not aim for peak performance, but a more relaxed, tempered, and sustained rhythm – pacing interaction and maintaining a sense of control, but also forestalling the onset of tiredness. Both conclusions are supported by the video study (Section 6.1), which not only notes the impact of tiredness, but also a rapid, yet tempered rate of interaction.

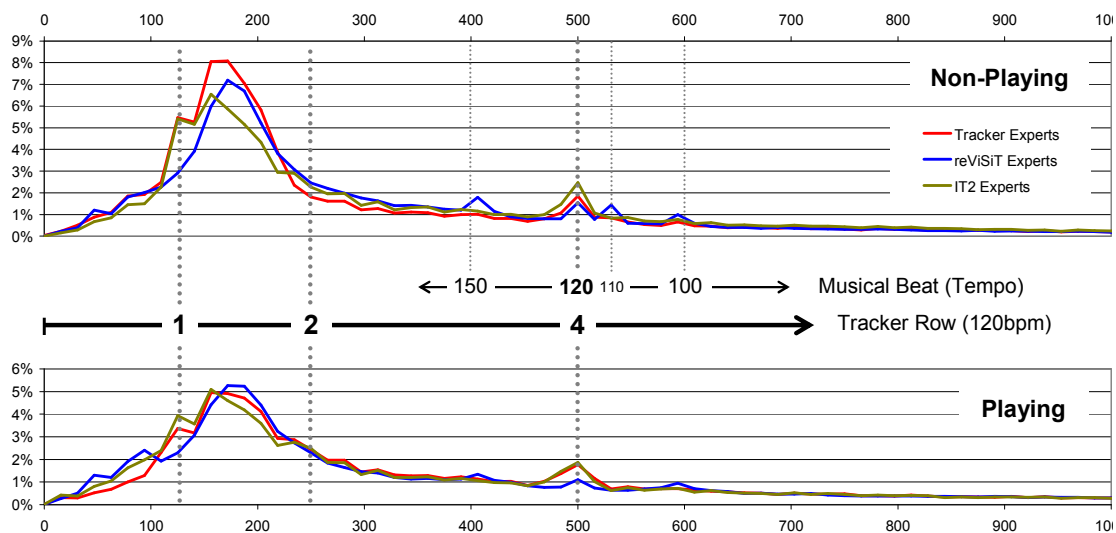


Figure 2 – Intervals between keys with and without playback (including non-typematic repeats). Histogram of inter-keystroke intervals (x-axis, in milliseconds), with guidelines for common musical tempo and tracker row intervals.

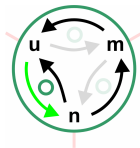
*rhythmic
cursoring*

In Figure 2, samples are taken of experts with differing tracker backgrounds, and include manually repeated keys, but not typematic repeats (when a key is held), to show the intervals between physical key presses. A similar peak around 150ms, followed by a long tail, is visible in the plot, but also accompanied by local maxima at several other intervals, which correspond to musical timings, notably the musical beat at the sequencer's default tempo of 120bpm (500ms), which occurs both during and outside music playback.² reViSiT experts, familiar with the more

² In Nash and Blackwell (2011), these results were presented including typematic repeats, which lead to large, additional spike at 20-30ms (typematic rate). Notably, it also accentuates the peak at 500ms (typematic delay). To scrutinise a potential link between musical tempo and non-musical interaction, the

complex handling and synchronisation of tempo in the host-plugin configuration show a more diverse use of tempi, with additional peaks corresponding to 100, 110 and 150 beats per minute.

In terms of flow, this is an indication of *action-awareness merging* – an implicit coupling of musical perception and motor action, where the environment influences the user’s behaviour. Such *entrainment* in music, such as the tendency of listeners to tap a musical beat, is been widely studied in psychoacoustic research (e.g. Clayton *et al*, 2005), but is here merged with program interaction, and shows that motor behaviour in trackers is subject to both conscious and unconscious influences (also showing the *interference* of visual and musical feedback in a *manipulation-driven* system; inset, see Figure 4-9). This interaction also has the effect of maintaining the continuity of physical activity in idle time between episodes of more focused editing,³ and may serve as an *epistemic action* (Kirsch and Maglio, 1994), where the cursor is stepped over musical material to aid mental simulation.



Finer divisions of the beat, corresponding to a single pattern row in the tracker (125ms), are also evident. The non-playing sample excludes note entry (which triggers playback of the note), but includes intervening cursor movement, which makes up most of these peaks. This behaviour corresponds to the entering of notes in near-realtime, specific examples of which were found in the video study and logs of other experienced users. The absence of similar peaks for *reViSiT* Experts might be explained by the more varied use of tempo, but may also reflect a skill associated with longer term mastery, not yet widespread in the younger program.

controlling time

Compared to live recording in the sequencer, the technique effectively extends a user’s command of the creative environment to the direct control of time. In terms of flow, the individual benefits from a greater sense of control, as the musical input rate can either be slowed to facilitate more complicated input, or accelerated to increase throughput. In this way, a user effectively self-regulates the balance of challenge and ability, allowing them to work at a natural pace that preserves a degree of musical continuity, without depending on realtime performance skills. Furthermore, the learning curve associated with tracking can be seen to reflect computer, rather than musical, literacy.

analysis presented here has been adapted to identify and filter typematic repeats from log data, producing a more accurate profile of physical user activity. The conclusions of the original paper, however, are still supported by the revised profile (Figure 2), in which the peaks remain evident.

³ Neurology research (Wickens, 2003) has linked high levels of dopamine in both motor activation and reward-mediated learning, contributing to an individual’s ability to maintain focus. As such, this habit in tracker users may represent an unconscious effort to self-regulate their level of engagement.

7.2 keyboard and program knowledge

transferring
knowledge from
other programs

Figure 3 plots the average times taken for keystrokes against the range employed by users across increasing levels of experience. Here, more experienced users are not only faster, but faster across a wider range of keys. Using a 1200ms latency as the threshold of “unfamiliar codes” (Card *et al*, 1980), novices show familiarity with less than 20 keys, while experts’ vocabulary is over 60 keys.

Less experienced users, even complete beginners, demonstrate some expertise with a limited range of keys, corresponding to simple commands, common to other software, such as basic cursoring (e.g. arrow keys) and data entry. Users with a little tracker experience show knowledge of a broader repertoire, executed slowly (possibly more deliberately), suggesting that keyboard layout and motor actions are yet to be fully memorised.

At higher levels of experience, the repertoire continues to increase, but more importantly, the proportion of those keys averaging faster times also grows rapidly. While users with the most experience (4) demonstrate familiarity with an additional 22.9 (36%) keys compared to other experienced users (3), the proportion of familiar keys below the threshold for skilled entry of “complex codes” (750ms)⁴ increases from 25.5% to 62.4%.

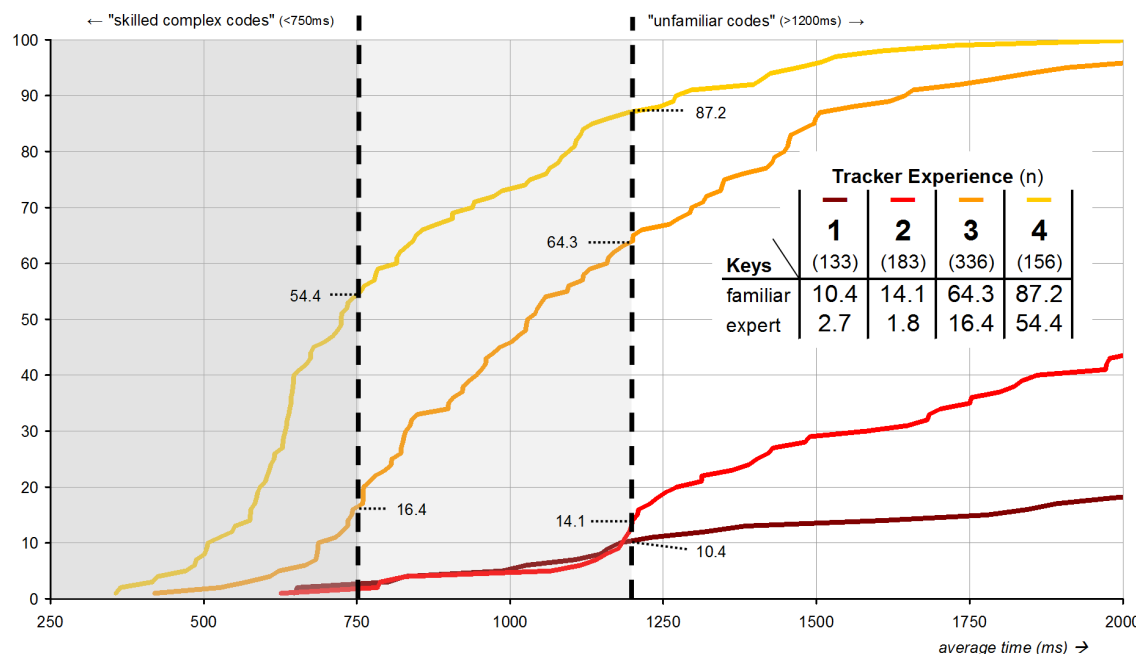


Figure 3 - Keyboard vocabulary. Number of distinct keys used, plotted against their average execution time (ms), across different levels of tracker experience (sample size in brackets). For example, experts show familiarity with 87.2 key commands, of which 54.4 are performed below the threshold of skilled use.

⁴ From Card *et al* (1980), which unifies several figures from Devoe (1967) measured using coded keyboards and matrices of keys, in which individual keys represent code words or commands. In the context of tracker interaction, this figure is applied to use of program shortcuts and keyboard macros.

The least experienced group of users (0) were excluded from the previous analysis.⁵ While these users tended not to persevere with the tracker experience (possibly intimidated by the learning curve discussed here), they also showed a propensity for input through the mouse and other modalities (MIDI, audio, etc.).

Programs based on GUIs, such as sequencers, favour bimanual interaction styles, involving one hand (typically, the user's preferred hand) on the mouse and one on either the keyboard or another device (e.g. mixer, MIDI keyboard).⁶ This style is also seen in the *Renoise* tracker (see Section 2.2.1; Figure 6-3), where cursor navigation, selection, and program settings are largely effected by the right-hand and mouse, but editing and playback by the left-hand and keyboard. While *reViSiT* (like *IT2*) is more exclusively designed for keyboard control using both hands, the video study shows a similar split in the responsibilities for each hand (see Figure 6-2), with the right hand rooted to the cursor keys. This consistency across different interaction styles, in music programs, may make it easier to move between them. At the same time, it highlights cursor navigation as one of the challenges facing new *reViSiT* users, who must learn to effectively use the keyboard, rather than the mouse, to get around the music and program. Figure 4 illustrates this transition, and the diminishing role of the mouse associated with greater *reViSiT* experience; moving from 64% mouse to 71% keyboard interaction. Use of the mouse requires visual inspection, such that the shift to the keyboard reduces the complexity of visual feedback, allowing greater focus on both the music and notation. The emergence of keyboard skills is further explored in the next section.

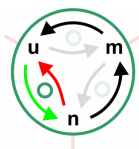
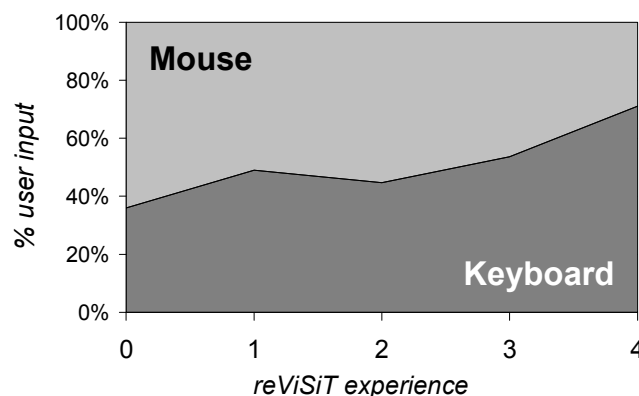


Figure 4 – Mouse and keyboard use by *reViSiT* experience



⁵ Users claiming absolutely no awareness of tracking (0) recorded too few keystrokes to support a reliable plot of the average; of the 41 users who provided more than 30 minutes of interaction, only 5,425 key presses were entered (0.2% of the total).

⁶ As Mackenzie (2003) notes, typical mouse usage breaks *Guiard's model of bimanual interaction* (Guiard, 1987), in that the preferred hand (typically, right) uses coarse movements to lead the non-preferred hand (left) and set the spatial frame of reference in which it operates.











7.3 a descriptive model of tracker interaction

generalising
user interaction

Keystroke selection varies between users, depending on their specific techniques, habits, and goals. Performance comparisons, such as the *keystroke-level model (KLM)*, are thus confounded, in creative design applications, by the absence of representative tasks or correct actions.

To enable comparisons between users, interaction styles, and programs, a *descriptive model* (Mackenzie, 2003) based on the broader context of actions performed in music editing was developed. Applying this model to tracking, a mask was defined for each command in the *reViSiT* program, using the contexts described in Figure 5, allowing individual keystrokes to be summarised and tabulated against user experience and interaction preferences, provided by survey data.

Figure 5
A descriptive model
of interaction in the
tracker, using context
flags to characterise
the general behaviour
of individual keys
(with examples)

SETTING		Changes settings or modes in the program
DATA		Enters data directly into the pattern (e.g. digits, text, notes)
AUDITION		Triggers incidental playback (notes, samples, live performance)
NAVIGATE		Navigates around the music (i.e. cursoring)
FOCUS		Navigates around the program (e.g. control focus)
PLAYBACK		Triggers song playback
SELECT		Uses block selection
CLIPBOARD		Uses the clipboard (cut, copy, paste, overwrite, etc.)
HELP		Accesses support documentation (built-in help)
EDIT		Flags a change in the musical data

Examples

<i>Play Song / Pattern / from Cursor</i>	⇒	PLAYBACK
<i>Note Entry</i>	⇒	EDIT DATA AUDITION
<i>Clipboard Copy</i>	⇒	SELECT CLIPBOARD
<i>Clipboard Cut / Paste / Mix</i>	⇒	EDIT SELECT CLIPBOARD

Figure 6 shows the breakdown of keyboard usage, for all users and across groups, characterised by different interaction preferences or levels of experience. Selected percentages and ratios are plotted in Figure 7 and summarised in Table 1, highlighting differences and progressions in interaction styles, dependent on user background.

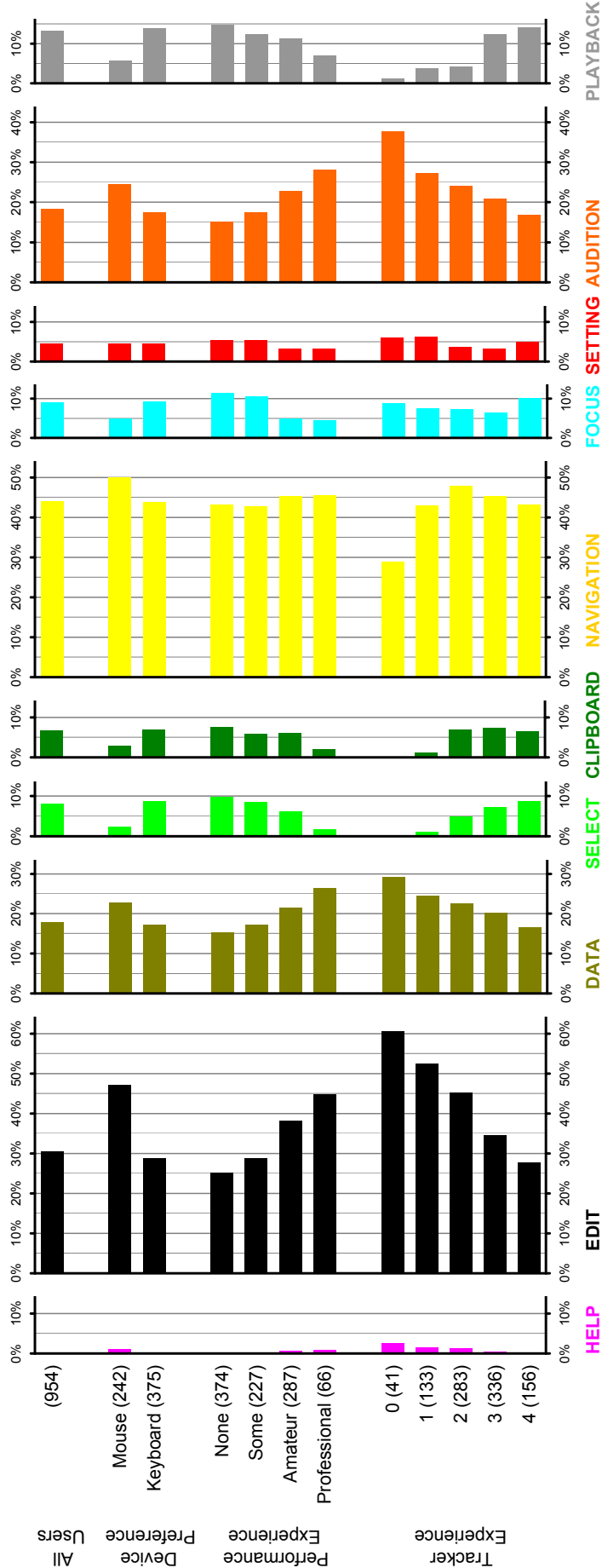


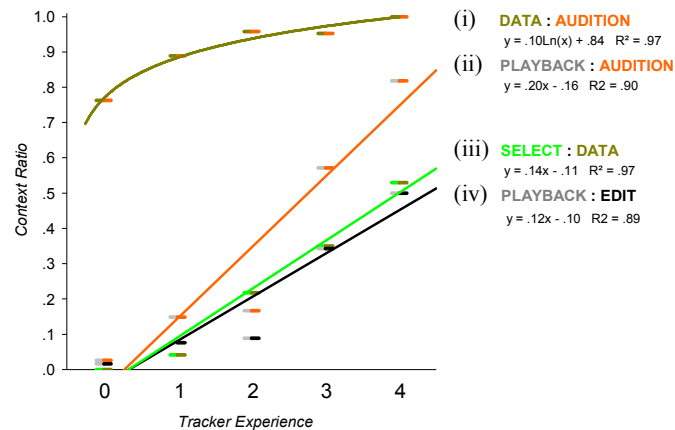
Figure 6 (above) – **Breakdown of keyboard use**, based on command context (see Figure 5), both for all users and across groups characterised by specific interaction preferences, as well as levels of tracker and music performance experience. (sample sizes in brackets).

Table 1 (left) – **Summary of keyboard use**, based on selected percentages and ratios of commands, as shown in Figures 5 and 7.

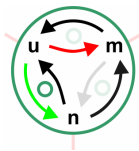
	All Users	Preference		Performance Experience				Tracker Experience				
		Mouse	Keys	None	Some	Am.	Pro	0	1	2	3	4
EDIT	30.4	47.1	28.9	25.2	28.9	38.2	44.7	60.6	52.5	45.2	34.5	27.7
PLAYBACK	13.2	5.7	13.9	14.7	12.4	11.4	7.1	1.2	3.8	4.1	12.4	14.1
AUDITION	18.4	24.5	17.5	15.2	17.6	22.7	28.1	37.7	27.4	24.2	20.9	16.9
DATA	17.9	22.7	17.1	15.2	17.3	21.4	26.5	29.2	24.5	22.5	20.1	16.6
SELECTION	8.1	2.3	8.7	9.7	8.4	6.2	1.8	0.0	1.1	4.8	7.3	8.7
HELP	0.2	1.1	0.1	0.0	0.1	0.5	0.9	2.4	1.5	1.4	0.4	0.1
(i) DATA : AUDITION	.97	.93	.98	1.00	.98	.94	.94	.78	.89	.93	.96	.98
(ii) PLAYBACK : AUDITION	.72	.23	.79	.96	.71	.50	.25	.03	.14	.17	.59	.84
(iii) SELECT : DATA	.45	.10	.51	.64	.49	.29	.07	.00	.04	.21	.36	.52
(iv) PLAYBACK : EDIT	.43	.12	.48	.58	.43	.30	.16	.02	.07	.09	.36	.51

Figure 7 – Trends in keyboard use, plotted as changing ratios in keyboard contexts, used to indicate:

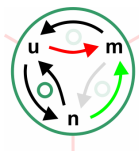
- (i) notation focus (vs. live music)
- (ii) feedback rate (*liveness*, see 4.2.4)
- (iii) feedback scope (song vs. note)
- (iv) editing scope (selection vs. note)



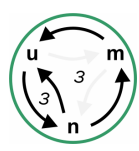
selection and clipboard use



increased use of playback



liveness in keyboard use



The **DATA : AUDITION** ratio (i) indicates the degree of editing that does not trigger an audition, and the shift from performance-like note entry to notation-based editing, as well as advanced control of dynamics (e.g. volume), spatialisation, and tracker effects. This more advanced editing quickly emerges with tracker experience ($R^2=.97$). An increased use of selections similarly represents more advanced editing, as indicated by the **SELECT : DATA** ratio (iii), which is also tied to experience ($R^2=.97$). Tracker experts edit faster and more efficiently, using selections to work at higher levels of musical abstraction (such as beats, bars, parts and phrases with multiple instruments), thus broadening the editing scope to include arrangement tasks (and the “big picture”, see 3.6). By contrast, live musicians, used to the performance capture model in DAWs, favour direct note entry over these more abstract control methods.

The **PLAYBACK : AUDITION** ratio (ii) indicates feedback scope, representing how often the wider song is played, in contrast to auditions of individual notes. With tracker experience, this ratio rises significantly ($R^2=.90$), from limited use of song playback by beginners, to near-parity with note auditions in experts. This is partly explained by the fewer auditions associated with the move to selection use, but also corresponds to fewer instances of users experimenting with instruments or melodies, before committing them to the notation. For experts, the *provisionality* of the notation supports sketching via destructive edits, removing the *premature commitment* of preparing a *performance*. This trend is also evident in the lower ratios of skilled performers, though greater musical knowledge may also reduce the user’s reliance on audio feedback.

In a similar regard, the final **PLAYBACK : DATA** ratio (iv) acts as a measure of *liveness* (see 4.2.4), indicating the changing ratio of playback and edit commands. Experts exploit this property of the notation to maintain a ‘live’ representation of the end product, the music (see Figure 4-10b). Like the previous ratio, this quantity correlates positively with tracker experience ($R^2=.90$).

Notably, these ratios only relate to keyboard interaction. A degree of equivalent functionality is offered by the mouse, as seen in novice use (see Figure 4). However, the more cumbersome use of drag-and-drop in the text-based pattern window and peripheral location of buttons (Play, Stop, etc., in toolbar) discourage their use, and make it difficult to maintain the same level of liveness available from the keyboard. The program's support for mouse interaction is instead designed as a teaching mechanism that exposes keyboard use and functionality to users more familiar with the use of the mouse, in sequencers or other music programs. To this end, clickable buttons, right-click context menus, and status bar messages supporting drag-and-drop operations always display equivalent keyboard shortcuts. The effectiveness of this strategy is underlined by the figures for keyboard-based program navigation (**FOCUS**), already prominent in unskilled use, and not significantly influenced by further tracker experience ($R^2=.01$).

Another indication of learning is provided by decreased use of help documentation, with greater tracker experience ($R^2=.95$). In the beginning stages, this documentation provides overviews, explanations, and tutorials, which later gives way to reference use for effect syntax and keyboard shortcuts, and which experts are ultimately able to commit to memory.

7.4 developing fluency in soundtracking

The development of virtuosity involves not only the use of advanced features, but also the integration of individual commands into fluid sequences of input. Expert tracker use, especially as regards the enabling of rapid edit-audition cycles, dynamically combines editing, cursoring, playback, and program control into passages of unbroken keyboard interaction. The matrices in Figure 8 illustrate the intervals between consecutive keystrokes, across increasing levels of tracker experience, broken down by context. At each level, two matrices respectively illustrate the fraction of intervals completed in less than 1 second, and the average speed based on all key transitions (in keys per second), accompanied by a summary and explanation of the results, supported by direct observations from the original logs themselves.⁷

⁷ The threshold of 1s is chosen to fall between those of “unfamiliar codes” (1.2s) and “complex codes” (0.75s), but also significantly below the typical threshold of mental preparation (e.g. $M = 1.35s$), in an attempt to isolate keyboard input that has been learnt to the point where it is executed with a minimum of conscious reflection (Card *et al*, 1980; see 7.2).

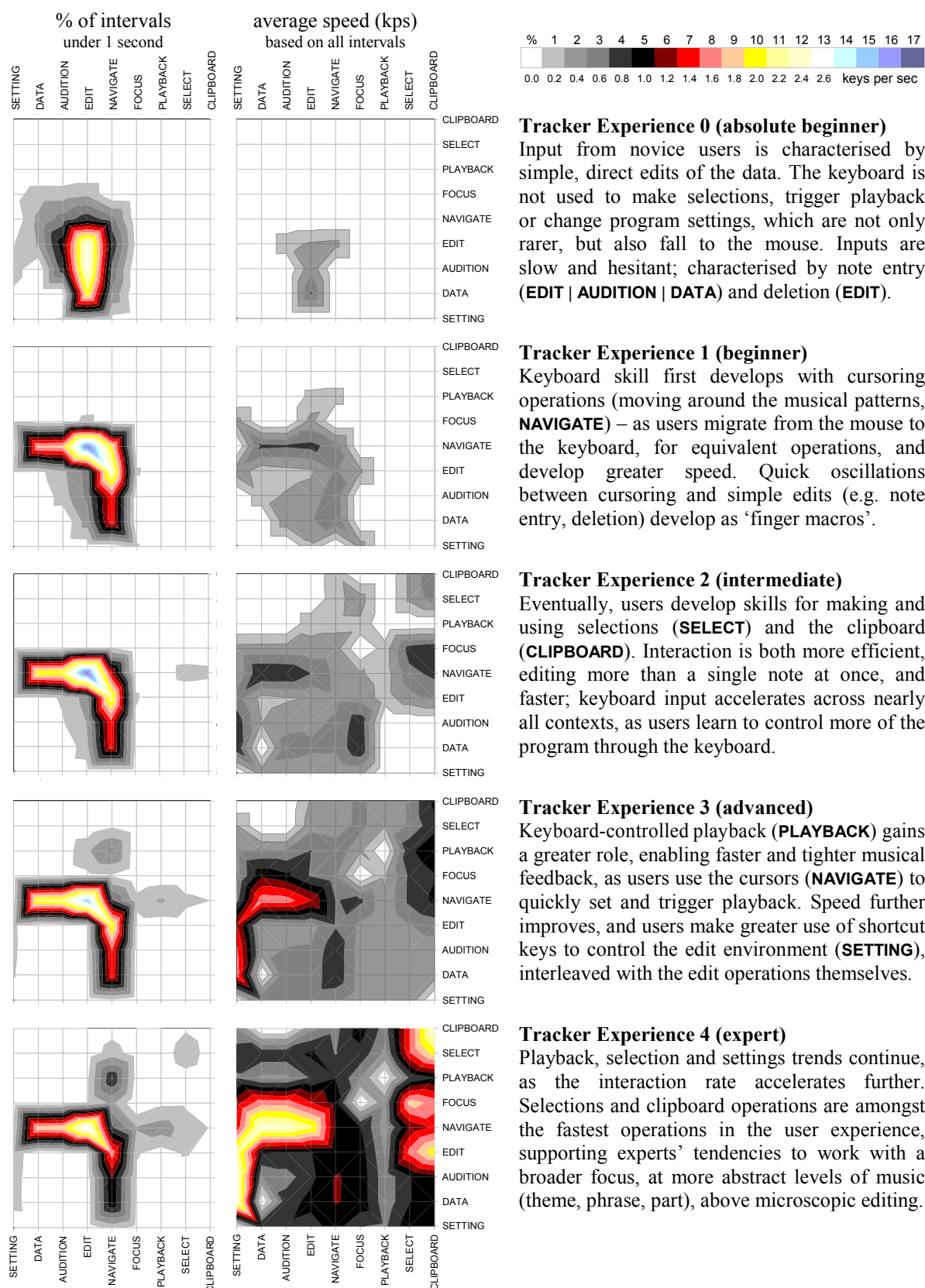


Figure 8 – Keyboard performance in the tracker, illustrated by inter-keystroke metrics across levels of tracker experience (approximate skill in brackets). Matrices indicate expertise within editing contexts: based on (left) the % of intervals under 1 second (x-axis – key 1; y-axis – key 2); (right) the average speed based on all intervals (in keys per second). A brief summary and explanation of results and trends, drawing on observations from user log data, is provided for each level of experience.

The development of specific keyboard skills is evident with each level of experience, providing a picture of how virtuosity is obtained with the tracker, through the keyboard. While even absolute beginners adapt to the virtual piano keyboard (see Figure 2-7), perhaps drawing on familiarity with the piano itself, more specialised tracker skills develop later.

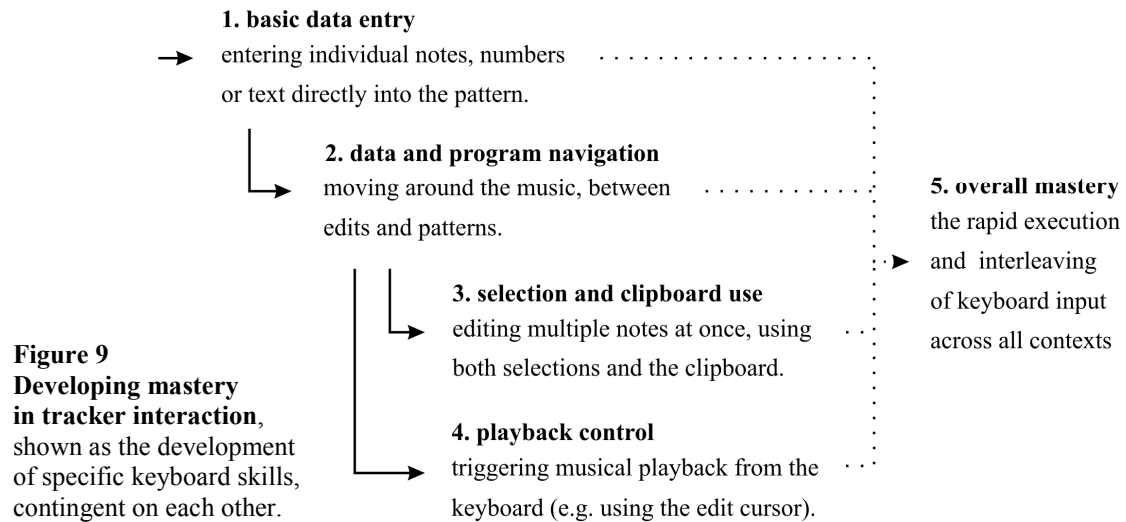


Figure 9 illustrates how new skills build on those developed previously – for example, cursor navigation is integral to both forming a selection and choosing where to start playback. The progression also corresponds to a gradual migration of interaction from the mouse to the keyboard (placing the cursor, drawing selections, and triggering playback). The final stage approaches mastery, where individual skills are not only combined, but quickly and seamlessly executed, across most contexts.

need for guidance and practice

Keyboard control of playback, especially, is fundamental in the tracker’s support for rapid edit-audition cycles, and the late emergence of this skill highlights a potentially serious usability issue – where functionality that may enable flow is not available to novice users. By themselves, the shortcuts and key sequences are straightforward, and draw on generic computer knowledge. Yet, while mastery of these skills might require extended practice (see Section 3.6), there is no obvious mechanism advertising or exposing them to the novice user.

Though reViSiT’s help system contains ‘Getting Started’ tutorials, which do emphasise the use of playback, there is little evidence they are effective, if used at all. The developmental progression observed here identifies the core skills in tracking, which could inform the design of more interactive tutorials that explicitly guide users towards more advanced use of the program.

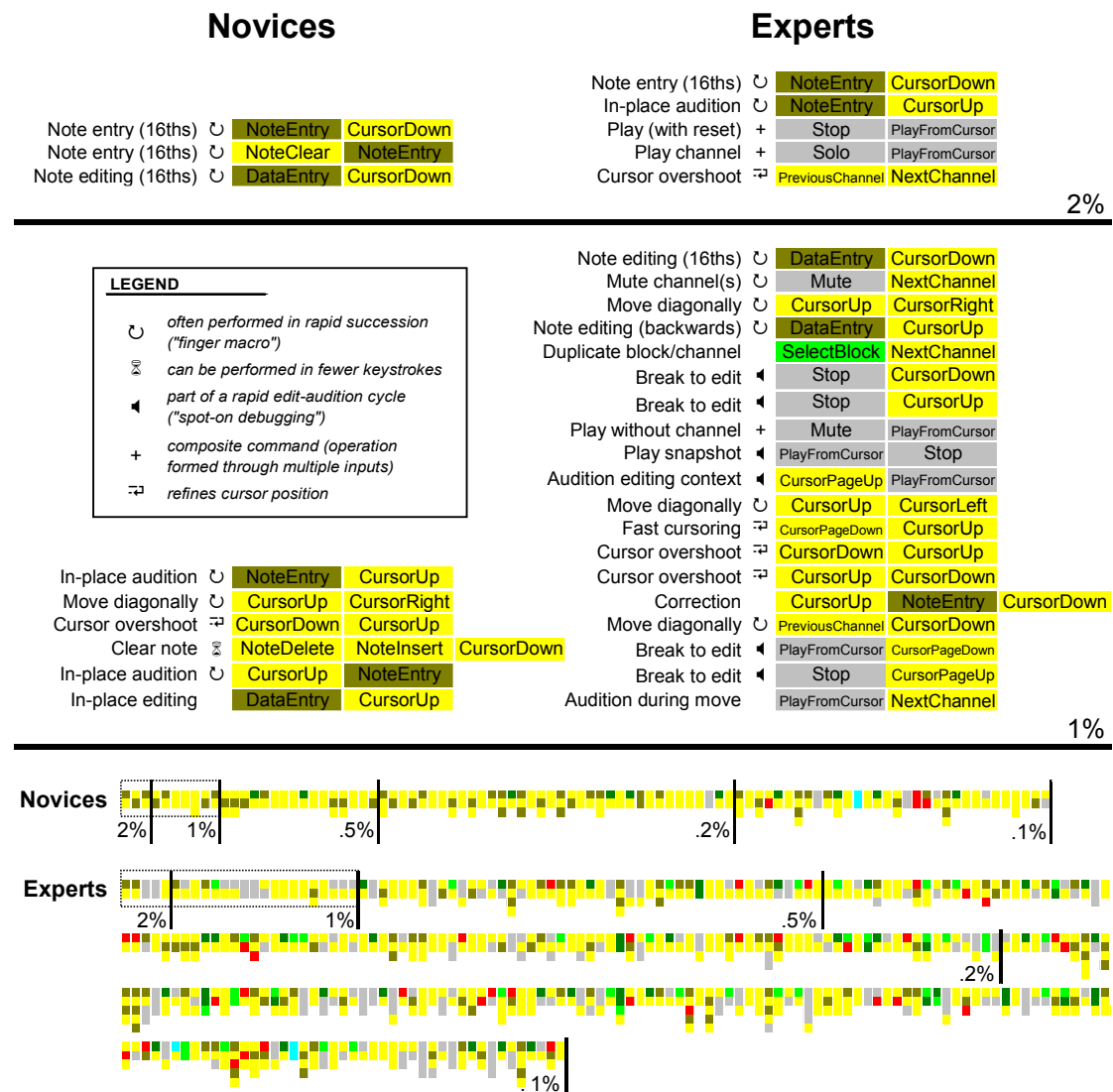


Figure 10 – Common sequences of keys, in tracker interaction. (top) Detail of the most common sequences, each constituting more than 1% of observed sequences, with a description of interaction and appropriate links to discussions in the main text; (bottom) General breakdown of sequences over 0.1%. Keys are colour-coded by context, as described in Figure 6.17.

*specific techniques
in tracker use*

Notably, the role of playback could be emphasised from an earlier stage of learning. Moreover, it might be possible to adapt the metrics used here to detect in realtime when a user is not making best use of the program, and could benefit from guidance – for example, monitoring average ratios of direct edits to selection / clipboard use, or overall edits to playback use).

Figure 10 illustrates common sequences of keys separated by no more than 1s, ranked by frequency (average percentage of user's sequences). The goal of this analysis was to identify sequences of keys that users are able to execute quickly, indicating that they have been well-learnt. Sequences represent possible examples of chunking in motor learning, where composite actions are

perceived atomically; that is, where the user thinks of a sequence of keystrokes as though it were a single gesture, allowing them to interact more abstractly, potentially in terms of program functionality rather than physical actions. As such, this represents a mechanism through which interaction is partially embodied, and processed unconsciously.

*keyboard sequences
as atomic gestures*

Figure 10 provides an overview of common key sequences, in abbreviated form, colour-coded by context (see Figure 5). 94 sequences were identified in common use by novices, focusing on data entry or cursor navigation, and knowledge transferred from similar interactions in other software, such as word processors and spreadsheets.⁸ By comparison, the 346 sequences identified in expert interaction also exhibit specialist knowledge, notably integrating playback control with other interaction contexts, as seen earlier (see Figure 8). Figure 10 also lists the most common sequences, each constituting over 1% of sequences exhibited by the average user. Sequences are annotated to describe the operation performed as a result, using icons cross-referencing discussions in the main text.

*near-realtime
composition*

The most common sequence for both novices and experts is a rapid oscillation between note entry and the down cursor. This has the effect of entering a sequence of notes spaced at regular intervals in the pattern, matching common musical note lengths, such as a quaver (2 rows) or crotchet (4 rows). In the video study (see Chapter 6), this behaviour allowed the user to step through the music, manually inputting at a rate approaching musical time.

“finger macros”

Such oscillations are an example of a wider phenomenon, sometimes called a “finger macro”, where sequences of two or more keys are repeated in quick succession, typically when the data being entered is itself repetitive. Other examples, in the tracker (denoted using a \cup symbol), include rectilinear cursoring, to approximate the most direct route between two points in the pattern, and the muting of multiple consecutive channels, to filter whole musical parts or instruments from the playback mix.

*in-place
experimentation*

A similar succession alternates between note entry and the up cursor, enabling in-place auditioning of notes.⁹ Unlike a finger macro, the up cursor is alternated with different notes, allowing the user to trial different pitches for a given position in the music. Here, the provisionality of the notation is sufficient to enable the user to experiment with their music using destructive edits.

⁸ The frequency of the ‘Clear note’ sequence likely derives from the subtly different cursor behaviour, prompting novices to string together three familiar actions, where more experts use a single key.

⁹ Note entry triggers the playback of the note.

*controlling
playback*

Three keyboard sequences are commonly used by experts to prepare and trigger playback. The keys for *Stop/Reset* (F8), *Mute Channel* (Alt-F9), and *Solo Channel* (Alt-F10) are adjacent to the *Play from Cursor* key (F7), enabling playback to be (re)started, focused, or filtered using single strokes or gestures of the hand, combining individual keyboard inputs (+). In the same manner, the composer in the video study was seen to reflexively press F8 before triggering playback, thus saving him from conscious reflection on the state of playback (or MIDI devices), and simplifying the process of getting musical feedback.

cursor overshoot

Holding a key down provides a quick, but imprecise, method to repeat keyboard input multiple times. For example, a user might move forward in the music by holding the down cursor, but is likely to undershoot or overshoot their destination. This situation explains the frequency of sequences, in Figure 10, that seem to backtrack or undo earlier actions (↶↷). Experts, however, are partially able to mitigate this using faster cursoring techniques that make use of the Page Up / Down keys and other navigation shortcuts (Home, End, Alt-Cursor, etc.).

*“spot-on
debugging”*

The specific sequences underlying the fast edit-audition cycle of the tracker, such as the “spot-on debugging” observed in the video study, are also highlighted in the list (◀). These include cursor navigation before and after playback. Having made an edit the user quickly moves the cursor back (*CursorPageUp*) and triggers playback. When stopped, the user promptly cursors back to the editing context. Very short playback episodes, less than a second in duration, were classified by the analysis as a single keyboard sequence; providing a momentary snapshot of the music, useful for feedback on harmonic content, orchestration, or to aurally gauge the music not currently visible on in the viewport.

Figure 10 highlights the centrality of the *Play from Cursor* command, in expert use of the tracker. Indeed, when playback is used by less experienced users, the tendency is to rely on playback mechanisms that play a whole section (*Play Pattern*) or the whole song (*Play Song*), befitting a working style where the composer spends longer periods editing larger musical building blocks, before auditioning them in more complete forms (see also section 8.3). This may simply derive from a lack of familiarity with the use of the *Play from Cursor* command,¹⁰ or it may indicate a more classical, analytical approach to composition, based on musical theory and traditional practices, as supported in other software, such as professional sequencers and score editors.

¹⁰ Even though its prominence in the UI and keyboard layout is equivalent to other playback functions.

The more synthetic approach of experimentation and exploration, associated with tracker use, only seems to emerge with experience, even despite the reduced requirement it supposedly places on notational literacy. It is likely that although the individual commands and sequences are simple to learn, their fluid and expert use requires a level of familiarity that only comes from extensive practice. In this way, an analogy is found in the skilled use of a musical instrument, from which it may be relatively simple to elicit a tone, but with which it is considerably harder to develop virtuosity (see both Section 3.5 and 3.6). To this end, this section has sought to identify aspects of interaction that correlate with such well-learned skills, and which facilitate fluid interaction and rapid feedback in the tracker.

Whether it is knowledge or experience of the program that a novice lacks, a program's support systems (online help, interactive tutorials) can use information about expert behaviour to provide advice or exercises for the user. For example, it may be of value to include interactive finger and keyboard exercises that focus on skills such as cursor navigation, to develop dexterity, motor skill, and coordination. Deliberate practice is an important component of developing musical expertise (Ericsson *et al*, 1993), and such provisions, based around keyboard interaction, may be a way to introduce it into computer music interaction. Ericsson *et al* assert that such practice is not inherently enjoyable, but programs may be able to integrate such exercises in more rewarding formats, such as a game that encourages improvement through competition (with oneself, *intrinsically*; or a community, *extrinsically*).¹¹

¹¹ The DOS tracker, *Fast Tracker 2*, had a built-in game called *FT Nibbles*, based on the classic *Snake* video game. Though it cannot be said that the programmers' intention was to develop a user's motor skills for tracking, the game consisted of very fast use of the cursor keys and rectilinear navigation, which could foreseeably translate to improved dexterity in the pattern editor.



chapter eight **focus and feedback in digital music**

This chapter looks at visual and musical feedback mechanisms in music software interaction. It looks at how a user's focus changes over the course of interaction, both between the tracker and host sequencer, and also between notation editing and music listening (Section 8.1). The mechanisms and use of musical feedback are examined and compared in each environment, especially in the context of editing activity, from which a measure of *liveness* is developed, based on the balance between playback use and both the duration and depth of editing (Section 8.2 and 8.3).

Section 8.4 subsequently explores how window and UI layouts influence a user's concentration, potentially dispersing their visual focus and requiring housekeeping that can interrupt and distract from music editing. Finally, in Section 8.5, previous discussions of both musical and visual feedback are brought together in the context of *FL Studio*, an advanced step-sequencer-based DAW that effectively combines a liberal use of windows, visual metaphor, and mouse interaction with focused editing and playback of short patterns of music.

8.1 activity profiles

Figure 1 (a) shows the focus and playback profiles for sessions of over 30 minutes, plotted against time (in % of session). Over the session, users spend an increasing majority of their time in the host, and an increasing amount of that time playing music. Intuitively, as music is created, more time is needed to audition it.

However, beyond the initial moments, no such increase occurs within *reViSiT*, despite the availability of song playback throughout the program. Instead, playback in the Pattern Editor levels off at roughly 20% of the user's time, with roughly another 5% listening time supported by other parts of the program. This suggests that playback in *reViSiT* is not used to listen to the wider musical context, but for shorter windows of musical feedback, simply to support editing. In turn, the gradual shift to host-based interaction could indicate the growing utility of the DAWs' higher-level arrangement and post-processing facilities (which can also be applied to music created in *reViSiT*). In this case, the increased playback would not only encapsulate listening, but also host-based editing, which also takes place in realtime (a Level 4 liveness *performance-based* system, see Figure 4-9; e.g. recording live audio, MIDI, or automation). Whatever its use, the corresponding drop in input activity for both environments tends to indicate that the host-based listening activities gradually replace, rather than support, more active editing interaction.¹

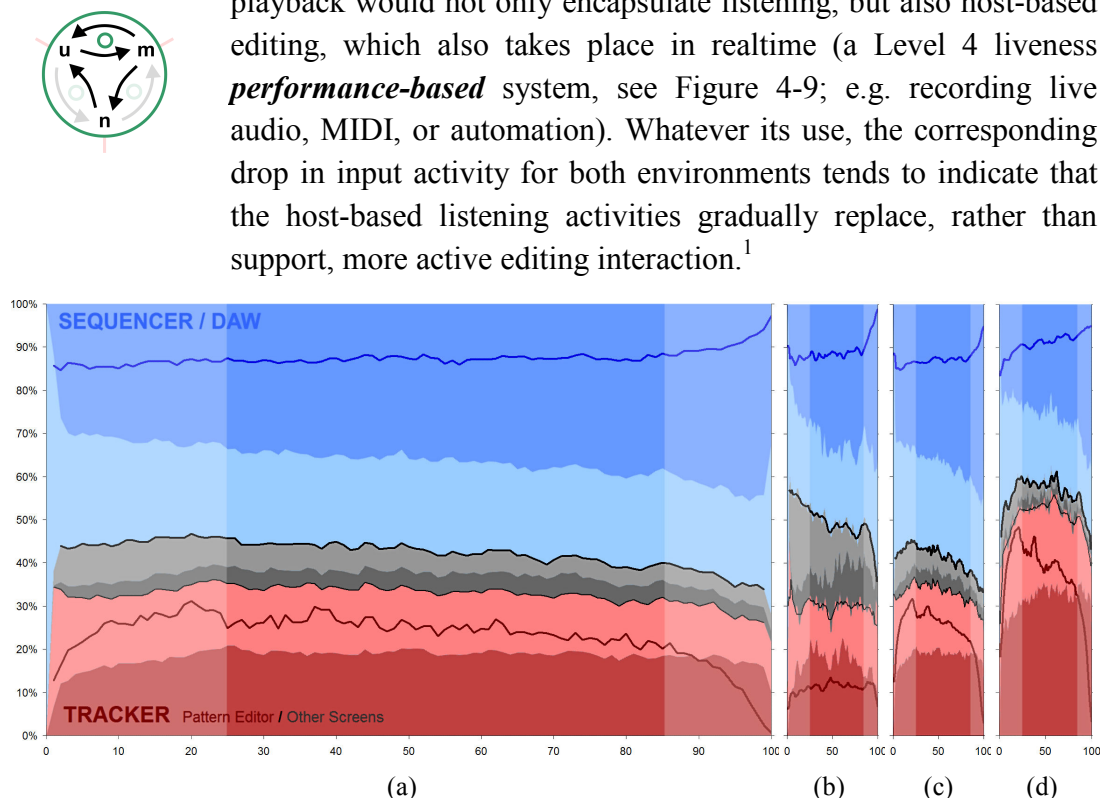
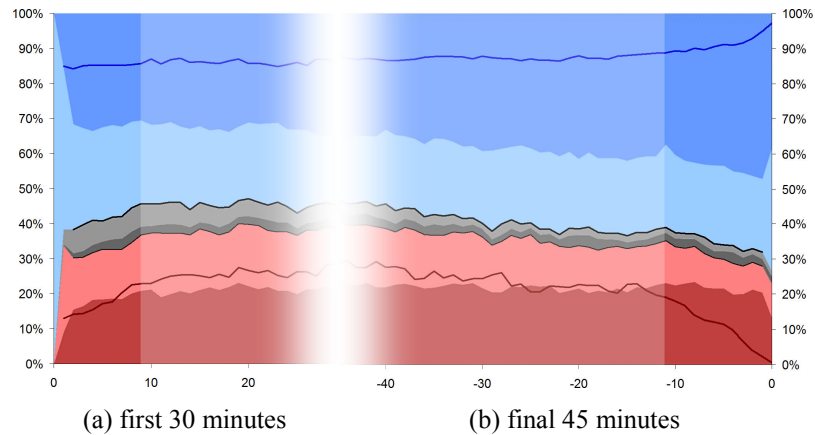


Figure 1 – Focus and playback profile of average session. An overview of the users' division of time between the *host* and *reViSiT* (including the *Pattern Editor*), as well as the proportion of that time spent listening to music (*shaded*), and curves plotting the corresponding level of input activity. Based on all sessions over 30 minutes in duration, plotted against time as % of total session duration, across different groups of users: (a) all users (1195 sessions, 175 users); (b) tracker novices (154 sessions, 68 users); (c) tracker experts (1041 sessions, 107 users); (d) *reViSiT* experts (276 sessions, 11 users).

¹ Note: Host MIDI input is not detectable by the *reViSiT* plugin, and is not included in the plots, but is assumed to be proportional to supporting mouse and keyboard interaction (e.g. used to initiate it).

Figure 2 – Focus and playback profiles at session start and end (legend as Figure 1) plotted against time in minutes. Based on sessions over 60 mins (508 sessions, 77 users):



stage theory in action:
preparation, creativity,
evaluation

The profiles are divided into three phases, characterised by changing trends in interaction corresponding to an initial period of *preparation*, a prolonged period of *creative editing*, and a closing period of *evaluation*. These phases, and the above trends, are evident in Figure 2, which illustrates the opening and closing minutes of longer sessions.

preparation
and start

Preparation lasts up to 9 minutes and is characterised by host interaction, slowly shifting to increased use of *reViSiT*. The amount of time spent in the Pattern Editor increases, as preparatory tasks in other parts of *reViSiT* and the Host (such as configuration of samples, tracks and workspace, or loading of songs and samples) are gradually completed. Activity supporting music editing (such as the management of instruments in other parts of *reViSiT*) continues, diminishing as the user settles on a musical palette and turns their attention to the music itself. Significant host-based playback (30%) is evident from the outset, indicating the likely existence of prior art, which the user plays to re-acquaint themselves, or possibly the use of realtime editing and recording features in the host. Users may take time to build momentum and find a rhythm once editing starts, contributing to less activity at the outset of the session. General observations of computer use suggest this can take up to 15 minutes (DeMarco and Lister, 1999), and may explain the continuing but less pronounced acceleration of input, in the subsequent period, in Figure 2, which appears to approach a limit during the first 20% or 20 minutes of interaction.

evaluation
and end

A final phase of interaction is visible in the last 10-11 minutes of sessions, characterised by diminishing *reViSiT* editing activity and a surge in host-playback. This shift corresponds to the users' progression to the later stages of the creative process (*evaluation, verification, elaboration, and refinement*; see 3.2) as they review their work, and make final edits, increasingly in the host, and at an audio (rather than musical) level.

delayed
gratification

Together, these two periods suggest that up to 20 minutes are potentially lost to activities other than musical creativity, where there is also less expectation of focused, engaging, and rewarding flow experiences. In this study, analyses are thus limited to sessions longer than 30 minutes in duration and, as appropriate, ignore the first 10 minutes of preparation.

In other areas, these shorter sessions help highlight specific usability issues and learning obstacles for new and novice users, and may identify ways to speed up the transition to hands-on editing. One obvious solution, to this end, is the provision of templates or presets that obviate the need for preparation, and which are becoming increasingly common in music, by way of bundled sample collections. In music programs designed for the consumer market, these libraries of pre-recorded sounds, loops and, longer musical phrases start the user at a more mature stage in the creative process, but enable creative tinkering with professional sounding results and minimal expertise. While they don't allow the flexibility of expression that artists and creative professionals might demand, they create a precedent for reducing the level of required preparation in music production. More professional pattern and loop-based programs – such as *Ableton Live*, *FL Studio*, and trackers – can also benefit from bundled sound libraries,² but also offer earlier gratification by focusing editing on shorter slices of music (see Section 8.3).

			All Users	Tracker		reViSiT	All Users
				Beginners	Experts	Experts	(normalised)
<i>users</i>			175	68	107	11	175
<i>sessions</i>			1195	154	1041	276	1195
Total	<i>playing</i>		59.49 ± 0.42	59.77 ± 1.05	59.50 ± 0.37	60.89 ± 0.64	51.61 ± 0.92
	Host	<i>focus</i>	56.89 ± 0.39	50.69 ± 0.61	57.85 ± 0.40	41.73 ± 0.49	43.67 ± 0.69
		<i>playing</i>	35.93 ± 0.39 (63% of focus)	32.33 ± 0.85 (64% of focus)	36.50 ± 0.39 (63% of focus)	25.04 ± 0.50 (60% of focus)	22.84 ± 0.63 (52% of focus)
	reViSiT	<i>focus</i>	43.11 ± 0.39	49.31 ± 0.61	42.15 ± 0.40	58.27 ± 0.49	56.33 ± 0.69
		<i>playing</i>	23.56 ± 0.18 (55% of focus)	27.44 ± 0.72 (56% of focus)	23.00 ± 0.18 (55% of focus)	35.85 ± 0.38 (62% of focus)	28.77 ± 0.58 (51% of focus)
	Pattern Editor	<i>focus</i>	33.75 ± 0.26 (78% of reViSiT)	29.99 ± 0.61 (61% of reViSiT)	34.26 ± 0.30 (81% of reViSiT)	53.05 ± 0.45 (91% of reViSiT)	32.05 ± 0.54 (57% of reViSiT)
		<i>playing</i>	19.35 ± 0.16 (57% of focus)	19.13 ± 0.61 (64% of focus)	19.39 ± 0.17 (57% of focus)	33.34 ± 0.39 (63% of focus)	18.02 ± 0.43 (56% of focus)
	Other screens	<i>focus</i>	9.36 ± 0.19 (22% of reViSiT)	19.32 ± 0.53 (39% of reViSiT)	7.89 ± 0.16 (19% of reViSiT)	5.23 ± 0.31 (9% of reViSiT)	24.28 ± 0.58 (43% of reViSiT)
		<i>playing</i>	4.20 ± 0.11 (45% of focus)	8.31 ± 0.57 (43% of focus)	3.60 ± 0.08 (46% of focus)	2.51 ± 0.15 (48% of focus)	10.75 ± 0.50 (44% of focus)

Table 1 – Summary of focus and playback, across user groups

Mean percentages and 95% confidence intervals based on the interquartile period (25-75% total duration) within average session, for four user groups with varying levels and types of expertise. Normalised figures for All Users are provided in the final column (see text).

² It was beyond the resources of this research to provide a sample library with the *reViSiT* distribution. However, this is partly mitigated by support for MIDI and soft-synth connectivity, and the widespread online availability of samples, as well as tracker songs that contain re-usable samples and instruments.

It is also important to note that each session only represents an extract of a creative process; a finished piece of music is typically the product of several sessions. As such, the activity represented is not that between a blank canvas and a finished work, but can begin or end with partially-completed material. Accordingly, the degree of relative changes and trends indicated by the data is expected to be greater across the wider creative process.³

The main body of interaction corresponds to a relatively stable period of editing, and gradual accumulation of musical material. Aside from the gradual trends observed above, the divisions of focus (and playback) otherwise remain in roughly constant proportion over the period. These proportions, however, vary significantly, depending on user background and expertise. In Figure 1, the sample is split between (b) novice and (c) expert tracker users, respectively; with (d) illustrating sessions from users with specific *reViSiT* expertise. Table 1 shows a breakdown of the average time (in % of session) users spent in each part of the system, plus the respective time in which music was playing. Figures are based on the interquartile period (25-75%) of the average session, in order to capture the main, productive phase of interaction, and ignore characteristics found only in the opening and closing moments of a session. The first four series correspond to the session profiles illustrated in Figure 1, targeting different levels of user experience.

Whilst Tracker Beginners spend significantly more overall time in *reViSiT*, compared to Tracker Experts (49.31% > 42.15%, $p < .05$),⁴ they spend significantly less time actually editing music in the Pattern Editor (29.99% < 34.26%, $p < .05$).⁴ This difference is largely attributable to the longer time novices spend in the tracker's other screens, learning about and editing instrument and song settings (19.32%). Experience with *reViSiT* outside the Pattern Editor (e.g. shortcuts, layout, function) allows experts to complete tasks more quickly. Because there is a low ceiling to the complexity of interaction in these parts of the program (and fewer paths to take), significantly less overall time is spent in them (7.89%, $p < .05$),⁴ which allows the user to devote more time to editing the music itself, either in the Pattern Editor or host program. This trend becomes more prominent when looking at users with experience of *reViSiT* specifically, rather than trackers in general. Over half these users' time is spent in the Pattern Editor (53.05 ±

³ This supposition is also supported by Figure 3, where the normalisation process increases the influence of users with fewer sessions (less prior art), and where such trends are also prominent.

⁴ Tested using a *one-tailed, unpaired Welch's t-test* (see Table 1 for sample sizes and confidence intervals).

0.45 %), and less than 10% of the total time in *reViSiT*'s other screens (5.23 ± 0.31 %). Unlike other users, *reViSiT* is the focal point of interaction (in focus for 58% of the time), yet a surge in host-based playback, at the end of the session, is still evident.

As a measure of the average session, Figure 1 (a) reflects the interaction of the users who contribute the most sessions, favouring the most prolific (and more experienced) users of the program, as can be seen by comparing the profile with that of Figure 1 (c). A fifth column of Table 1 uses the same dataset as the first series, but normalised to users (i.e. each user's sessions are averaged before the profile is summed with that of other users). The corresponding normalised profile is pictured in Figure 3. With this process, a novice with only one session influences the result as much as an expert with several (at the cost of increased variance from summing of fewer profiles, themselves based on fewer sessions). The resulting figure is a measure of the average user (rather than the average session), a significant proportion of whom are new to *reViSiT* (94%) and tracking (39%), thus favouring novices and providing insight into earlier stages of learning, as confirmed by the similarity with the earlier Tracker Beginners session profile (Figure 1 (b)). Similarly, these results also emphasize these users' earlier stage in the creative process, with the corresponding profile in Figure 3 initially showing less host playback (due to the lack of prior art) and more preparatory activity in the tracker (outside the Pattern Editor), as well as less signs of a surge in host playback, at the end of the session, that characterises the conclusion of a creative process.

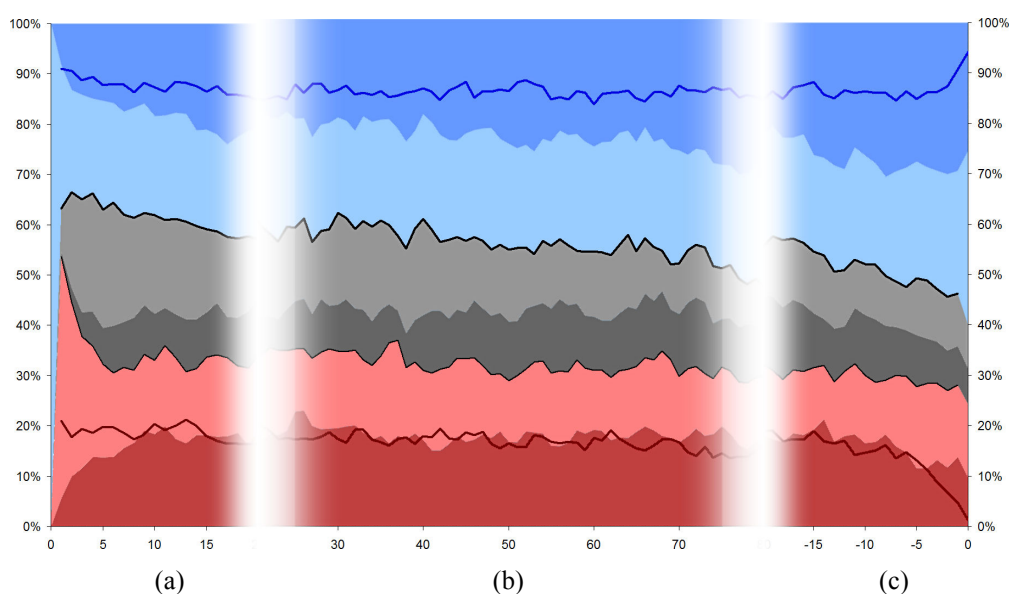


Figure 3 - Session profile of average user (legend as Figure 1), based on sessions over 30 minutes duration (175 users, 1195 sessions), normalised by user: (a) first 15 minutes; (b) second and third quarters (25-75%); (c) final 15 minutes.

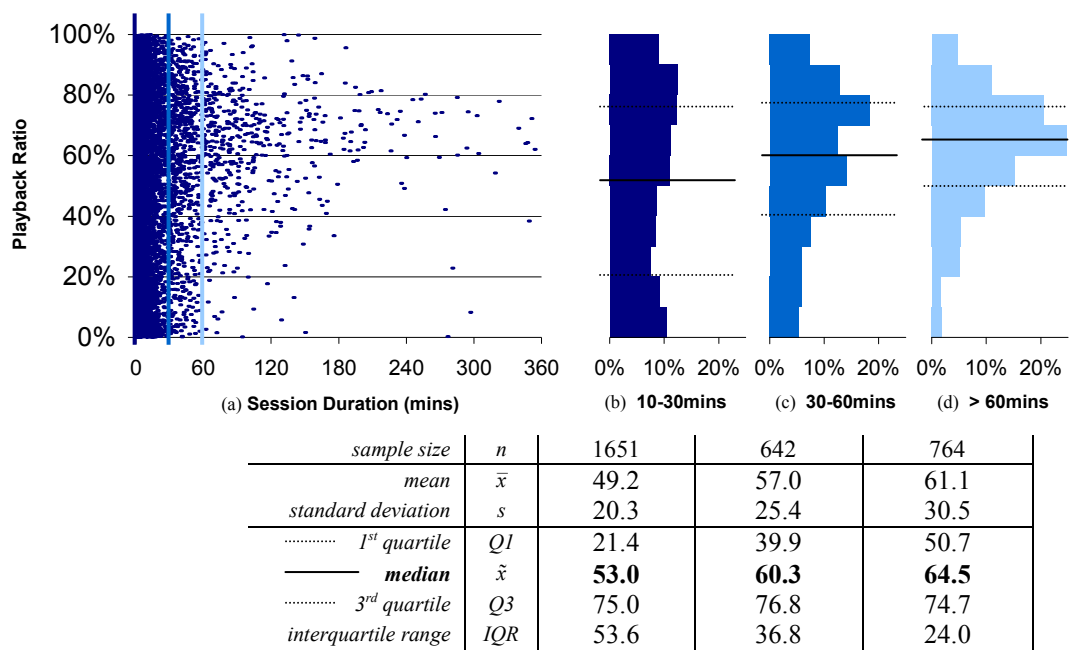


Figure 4 – Proportion of time spent playing music. Percent of activity with music playing, plotted as (a) a scatter plot against session duration, with histograms (and summary statistics) for playback use in sessions (b) under 30 minutes, (c) 30 to 60 minutes, and (d) over 6 minutes.

audio feedback
threshold

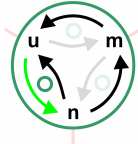
Despite varying focuses, user backgrounds and programs, the overall percentage of time spent playing music remains surprisingly uniform; averaging 60% of the time a user spends active.⁵ Though expertise in either the tracker or sequencer can be seen to increase playback use in the respective environment, it is generally balanced by less playback in the other.⁶ The division of focus between the two environments also does not appear to affect the overall use of playback ($R^2=.04$).

Figure 4 plots the proportion of time in which playback was playing, against session duration. Figure 4 (b) shows varied use of playback in short sessions (< 30 minutes), with a near uniform distribution of playback ratios – from negligible use of playback (associated with visual editing, learning, or exploring the UI) to near-continuous playback (associated with live editing, recording, ‘macro-listening’ to a song, or browsing different songs, possibly from other artists). In longer sessions (Figure 4 (c & d)), associated with more productive editing activities, these extremes all but disappear, and tend towards an increased, but balanced use of musical feedback. In sessions over 60 minutes, playback is active

⁵ Normalised by user, this proportion drops to 52%, as the influence of users with fewer sessions (less perseverance) is increased. Beginners with more sessions, however, spend 60% of their time listening to audio, which may signify an early rise in playback use, after the first session, or 30 minutes.

⁶ For example, sequencer experts (expected to use reViSiT in a supporting role) did demonstrate a higher proportion of playback use in the sequencer itself (74%), but balanced by a lower use of playback in reViSiT (55%), which still culminated in an overall average of 64%, in line with the trend).

domain feedback
and liveness



for roughly two thirds of interaction time ($\bar{x}=64.5\%$), with over half exhibiting playback ratios of between 50% and 75% ($IQR=24.0\%$). The tendency towards this level of playback is illustrated by the normal distribution evident in Figure 4 (d), and supported by an interquartile range test for normality ($z_{Q1}=-.68$; $z_{Q3}=.67$).

As domain feedback, greater use of audio playback instils greater liveness in the user experience (see Section 4.2.4), and facilitates the editing of visual notation (see *flow interference*, Figure 4-9). In music, hearing the actual music, rather than interpreting an abstract representation of it, leads to more direct involvement (Leman, 2008). The feedback motivates a user not only by providing early gratification for effort already expended, but also the impetus and guidance for further interaction. Duration, in this context, can thus be used as a simple, but effective indicator of activity where motivation is maintained. The predominantly non-professional and non-social use of the program also suggests that such motivation is likely intrinsic to the user experience, conducive to both enjoyment and the conditions for flow. Figure 4 suggests that prolonged interaction, as an indicator of motivation, increasingly depends on high availability and increased use of musical feedback.

Several factors also serve to limit the use of musical feedback. While sequencers support realtime entry and manipulation of music (e.g. recording), other editing operations are not connected with playback, especially where they concern manipulation of a visual notation. This asynchronous mode of editing is even more common in the tracker, where edits are made and then auditioned sequentially. As such, it becomes unnecessary (and even difficult) for a user to have music playing continuously. Indeed, such a scenario may not be desirable. Unlike listening for pleasure, musical feedback that supports editing is more broken, repetitive and strenuous, both mentally and physically. In studios, engineers can suffer physical discomfort (*ear fatigue*) as a result of sustained listening to music at high volume levels. More generally; focused, engaged interaction and concentration has a tiring effect on the individual. In the video study, the composer was observed to adjust his way of working to drop his work rate and take longer breaks to combat tiredness (see Section 6.2).^{7,8}

⁷ His principal technique was to break from the rapid tracker edit-audition cycle, and move to the sequencer for more relaxed listening, to longer excerpts of the song. Though this has the effect of increasing the net use of playback, the less broken, more structured, and polished nature of the audio feedback ultimately places less strain on the ears.

⁸ Another question, not explored in this research, concerns how the accumulation of material, and increasing time spent reflecting on it, impacts or encroaches on subsequent editing and the creation of new material. How does an individual's satisfaction with their creativity balance with their confidence to maintain it? Are users able to recycle the creative process, or do they quit while they are ahead?

This section correlates the first and last phases in the session profiles with the respective extremes of the creative process, as defined by stage theories of creativity (see Section 3.2). These stages (*preparation*, *evaluation*, and *elaboration*) are distinguished by conscious activity, and are the easiest to identify and observe. The intermediate unconscious stages of creativity (*incubation*, *intimation*, and *illumination*) are harder to delineate. In user logs, passages of interaction are sometimes characterised by lulls and unfocused sketching that precede bouts of high-energy productivity, followed by extended listening, but it is difficult to determine, with any reliability, how this progression relates to these stages.

The variation between individuals means that an average taken over users and time will obscure these details, which are better revealed by detailed studies of individuals. The composer in the video study, for example, mentioned long periods of simply listening to his music at length, distinct from editing activity. These periods, which he calls “macro-listening”,⁹ can easily last over an hour, where he simply places the music on cycle and sits back. They not only serve for analysis of his work, but to refresh his feel for the music after an absence from editing, or simply because he enjoys the music. As “time away” from active editing, this passive phase of interaction thus serves to incubate new musical ideas and artistic decisions ahead of editing, but in an explicitly disengaged, unhurried and unfocused manner. It highlights an example of *incubation* later in the creative process (after the production of an “intermediate form”) and a role for less conscious reflection during *evaluation* stages, such as *verification* and *refinement*, lending support to models of musical creativity as iterative and parallelised creative processes (e.g. Webster, 2002, Knörig, 2006; see also Section 3.2).

It is also useful to consider Graf’s (1947) description of the composition process (see both Section 3.2 and 3.5); as a gradual transition from less conscious creativity, supported by sketches, to increasingly critical thought and reflection, culminating in the final score. This progression may be evident in the sessions, the gradual shift from tracker-based editing to increased use of higher-level editing, song playback, and post-processing in the sequencer.¹⁰

⁹ And which he distinguishes from “micro-listening”, the very short (<2s) bursts of audio feedback used to support editing. See Sections 8.2 and 8.3 for more detailed analyses of playback use in music software.

¹⁰ This is supported by later analyses (see section 8.3) that reveal tracker editing to focus on fast, provisional editing cycles, characterised by frequent, short episodes of musical feedback, conducive to sketching and experimentation with new ideas, facilitating early-stage creativity. Other studies have also found the main value of sequencers and other music software arises during the later stages of the creative process (Blackwell and Green, 2000; Smith *et al.*, 2009).

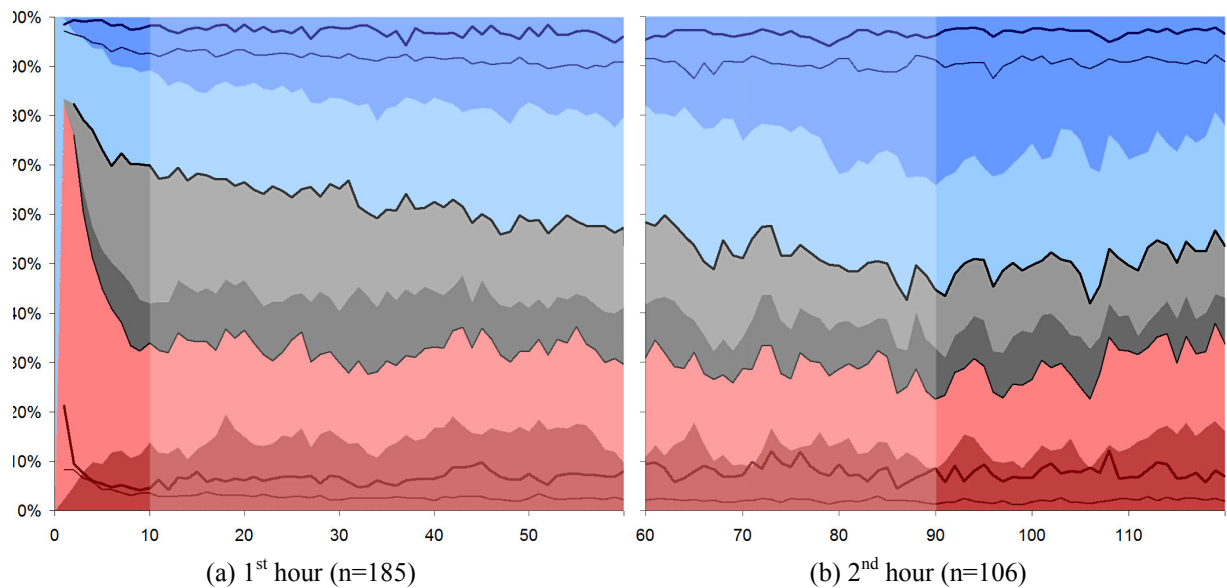


Figure 5 – User profile over time. Concatenates users’ sessions to illustrate the first 4 hours of interaction, showing focus splits between the **host** and **reViSiT** (and **Pattern Editor**), as well as the proportion of that time spent listening to music (**shaded**), and curves plotting the corresponding level of input activity.

*user development
over time*

Figure 5 concatenates each user’s sessions to provide a picture of the average user’s first 4 hours in the tracker, and their initial exposure to the program.¹¹ As such, it illustrates not only the newcomer’s early development of skills and practices, but also coincides with the start of a creative process, not confounded by the continuation of previous work evident in session profiles.

Initially, the user spends the majority of their time in *reViSiT*, exploring, experimenting, and learning the program. In the first 10 minutes, this exploration is focused on the Pattern Editor, but soon split with other supporting screens that govern the loading and editing of instruments, and are a pre-requisite of music editing. Focus on the Pattern Editor remains stable, at around a third of the user’s time, while use of other parts of the program diminish, reducing the overall proportion of time spent in *reViSiT* linearly over the next 90 minutes ($R^2=.87$), towards parity with the host. After this point, the majority of a user’s interaction in *reViSiT* is focused on the Pattern Editor and music editing.

Total playback rises quickly from zero, approximating a logarithmic curve (Figure 6, $R^2=.88$). After 20 minutes, playback in *reViSiT* stabilises, occupying just under a third of the user’s time, but sequencer playback continues to rise linearly for the next 90 minutes, accompanied by a corresponding shift in focus, from the

¹¹ The overlapping sections of the (b) 2 and (c) 4 hour samples are not shown. When overlaying the duplicated sections, the sole discernable distinction is increased noise in the samples that are based on fewer users. For this reason, the role of these omitted sections is served by the displayed profiles, together providing an accurate representation of interaction trends in the first 4 hours of *reViSiT* use.

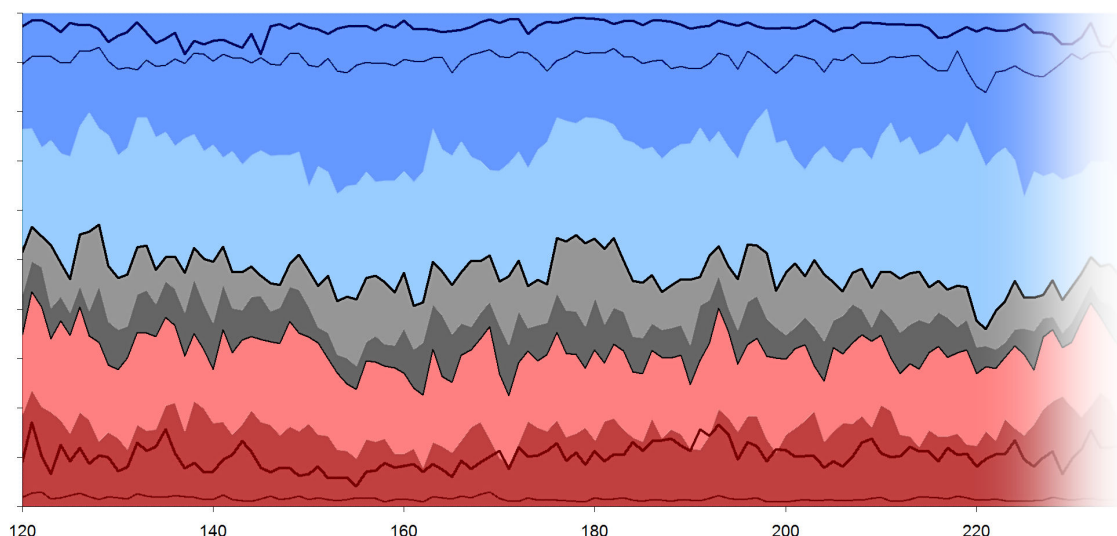
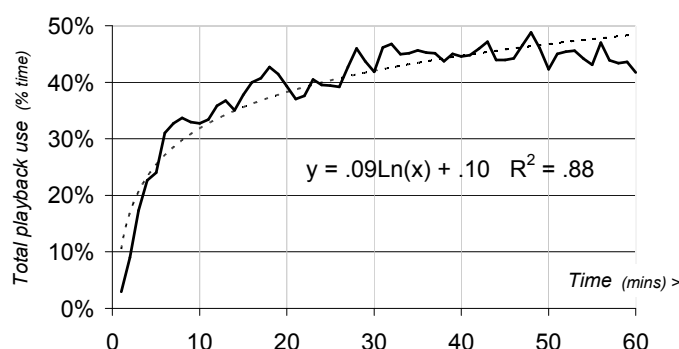


Figure 5 (contd.) – (c) 3rd and 4th hours (n=67)

tracker to the sequencer. Here, as the skill with *reViSiT* matures, a user's attention extends to combined use of the tracker and sequencer environment, such as the application of the sequencer's post-processing facilities to music created in *reViSiT*.

Figure 6 – Total playback over time (1st hour of activity)



After roughly 90 minutes, focus and playback use approaches that demonstrated in the sessions of tracker experts (see Figure 1 (c)), which may indicate an important milestone in the learning curve. At the same time, as much as this may reflect skill acquisition, it likely also reflects the gradual build up of a sample collection, upon which more experienced tracker users can draw – thus saving time on preparatory and peripheral tasks, enabling users to focus on editing music in the Pattern Editor.^{12,13}

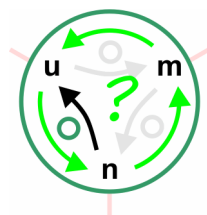
¹² This explanation is supported by the higher percentage of time spent in the Pattern Editor, by Sequencer Experts, during early interaction (76% of total time in *reViSiT*). These users have no tracker background, but likely have a sample library. *reViSiT*'s supporting screens are also based on concepts and interaction styles seen in DAWs (e.g. software samplers), which may support the transfer of their expertise.

¹³ Music programs are increasingly bundled with large libraries of samples, presets, and templates to help new users get started, and jump straight to music editing. Unfortunately, a similar provision was beyond the resources of this research, such that *reViSiT* is not strictly usable "out of the box" (see also Footnote 2), which may have influenced the initial appeal of the program, and the wider retention of users with less experience or knowledge of trackers.

Moreover, while these figures might indicate the assimilation of the broad tracker concept, deeper and more advanced expertise, such as motor and keyboard skill takes longer to develop, only approaching that of users with general tracker experience in the fourth hour of interaction ($\bar{x}_{180-240}=11.25 \text{ cmds/min}$) – and still a long way from the mastery and interaction rate of users with specific *reViSiT* (or *IT2*) tracker expertise (see Section 6.2.1).

Minor fluctuations in the focus average are also subsequently visible, in Figure 5, in which an oscillation between the tracker and sequencer is visible, in cycles of roughly an hour. Based on the earlier session profiles (e.g. Figure 1), this might indicate periods of tracker-based editing, followed by host-based refinement and evaluation, and thus correspond to iterations of the creative process. The lack of any significant correlation between the time spent in the host and the input activity, may also suggest that the sequencer is predominantly used for listening. However, the increased noise in these smaller samples, as well as the lack of detailed data on how the sequencer is used during these periods, makes it difficult to explore this hypothesis.

8.2 measuring liveness



liveness in tracking

In a music program, timely audio feedback is perhaps even more critical than visual. Analysing keystroke categories (see 6.2.2) showed that users with different amounts of tracker experience differed in their use of playback. Novices tend to audition their music from the beginning of the piece (*F5*) or phrase (pattern, *F6*), while more experienced users audition shorter passages at or around the editing cursor (*F7*), developing a rapid edit-audition cycle, with edit commands interwoven between playback of a single tracker row, the beat (4 rows) or the bar (16 rows).

Though the net use of playback is similar in both trackers and sequencers (see section 8.1), Figure 7 highlights a difference in individual episodes of playback, in each environment. Analysing the lengths of playback episodes, it is evident that trackers support a tighter edit-audition cycle. Whilst greater experience leads to faster feedback cycles in both programs, the overall distribution of playback episode lengths shows how sequencer playback is heavily quantised to musical bars, as illustrated by the spikes in Figure 7, during shorter auditions. The mode of tracker episodes is a duration of 0.5s (1 beat at 120bpm), the peak of a long, smooth tail that shows varied and flexible use of playback, with slight shelves corresponding to the musical bar (2s / 16 rows in 4/4) and tracker pattern (4s / 64 rows). By comparison, sequencers show a

strong tendency towards whole bars and longer phrases – at 2s, 4s, and 8s (1, 2 and 4 bars at 120bpm, 4/4), and also 10s, 20s, 30s, 45s, 60s, and 90s, for workspaces using digital timecodes.

Figure 8 illustrates playback use in the tracker and host sequencers, highlighting not only the quantisation of sequencer playback episodes, but the *premature commitment* that requires a user to set a duration, which is then doggedly maintained throughout the session. By comparison, few such trends are visible in the tracker plot, except wider tendency towards shorter playback snapshots, with almost half (48.4%) the episodes under 2 seconds in duration (compared to 34.6%, in the sequencer).

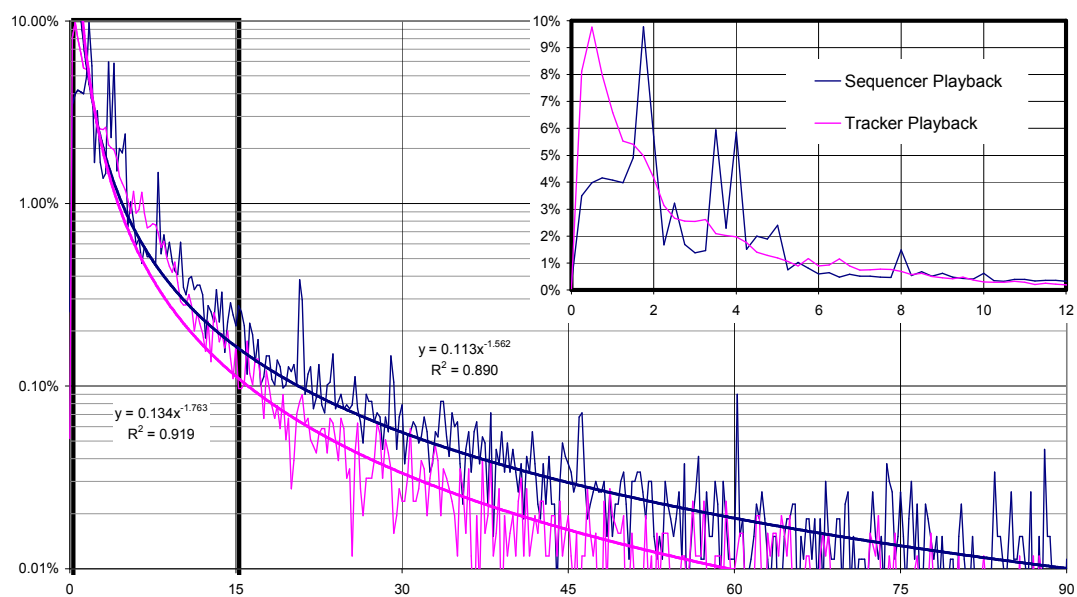
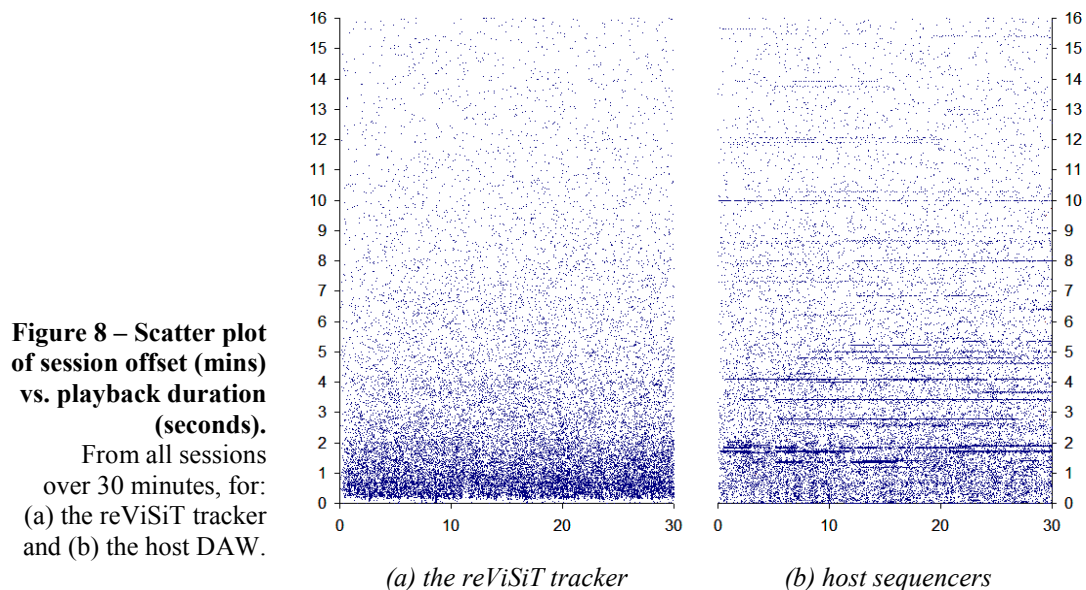


Figure 7 – Durations of playback episodes, plotted against the frequency of their occurrence, using logarithmic scale up to 90 seconds, inset with detail under 12s duration using linear scale.



This disparity with tracking can be explained by the provision of cursors in the UI, which indicate where the focus of work is. Both programs maintain a playback cursor, but while the keyboard focus of tracker programs provides a constant edit focus, no fixed focus or context is supported by the sequencer's mouse pointer, which can affect any moment or property in the music, across any window. Auditioning recent sequencer edits thus requires a more involved reconfiguration of the playback cursor, which is then typically snapped to the nearest bar or beat. That the sequencer maintains a persistent playback, rather than edit, cursor is an example of the different emphasis, as compared to tracking, in line with the system archetypes defined earlier in Section 4.2.3.

*reflective vs. reflexive
use of playback*

The need to step out of editing inhibits frequent and fluid use of musical feedback, encouraging more sparing use of playback in the sequencer. In Figure 7, both distributions fit an inverse power curve (Sequencer, $R^2=.890$; Tracker, $R^2=.919$), in which the higher exponent in the sequencer curve reflects a tendency towards longer playback durations. This trend is notable in playback over 10s, indicating sustained listening to the song. While this may be attributable to live recording and the capture of longer performances, the concentration of sequencer playback at the backend of the session (see section 8.1) points to more reflective evaluation (or post-processing refinement) of previously-recorded material. In this regard, the added precision and preparation supported by the sequencer's playback control may suit the more conscious thinking styles that characterise late-stage creativity.

In the tracker, playback control is more reflexive. Evidence for this comes from the frequency of extremely short audition clips, truncated to less than 100ms, before they can usefully provide feedback. These false starts suggest users have a well-learned, automatic mechanism for triggering playback after short edits, only reflecting on the appropriateness of the action retrospectively. Rapid edit-audition iterations build momentum so that this double-take can occur when the user switches to a more reflective mode of listening, requiring more forethought or preparation.

playback caesura

These slightly longer auditions act as punctuation marks in editing, like a typist using a full stop, a programmer using a semi-colon, or even a musical *caesura*. Focus, however, is maintained, through the tracker users' tendency to hover over the *Stop* key (F8), to truncate auditions the instant they have served their use. The composer in the video study (see Chapter 6) described this approach as "spot-on debugging", allowing him to jump straight back into editing when he hears something of interest. As a

*bottom-up
composition*

consequence, the tracker user remains engaged, and their attention focused, in the short episodes of playback linking different periods of more active editing.

In all, three distinct modes of listening emerge in tracker use, supporting a bottom-up approach to composition, hierarchically increasing in temporal scope: *note*, *phrase*, and *song*. At the lowest level, notes are entered and edited with support from very short auditions of single beats, but still within the wider harmonic context of the piece. These fine edits are sequenced and layered, culminating in short musical phrases, of one or more bars, which are auditioned as a whole. Finally, these bars, phrases and patterns are sequenced and layered to form a song, supported by longer, more reflective interaction. This approach immerses the user in the detail of the music, in contrast to analytical, top-down composition styles, built on knowledge of musical structure and theory.

*engagement
and intimacy*

Feedback from one user, with significant past experience with Impulse Tracker, goes as far as to describe a closer physical proximity to the computer itself, when interacting with reViSiT:

I felt so familiar with this program, like it has always lived in my computer, and it kind of pulled me forward to sit in front of the screen and focus on overcoming the limitations of the host.

Other users likewise describe a greater sense of intimacy, one likening the user experience to that “more like an instrument”, attributed to the speed and spontaneity of interaction. Other former DOS tracker users also commented on the lower level of engagement in Windows and sequencing, which frustrate fast interaction and focus through the need to switch between windows and input modes or devices. In this sense, greater liveness (as *direct and immediate feedback*) is also a critical factor in the *sense of control*, leading to more reflexive use of a program, but hindered by the more reflective playback control in sequencers.

the “big picture”

The overall tendency for tracker experts to focus on finer details and shorter sections seemingly contradicts other findings that observe expert musicians’ ability to focus on the “big picture” (Chaffin and Lemieux, 2004). However, this finding should be interpreted as a reiteration of their improved ability to retain the big picture in mind, without recourse to visual or aural feedback. Thus, the lack of these scaffolds has more implications for developing users, less able to visualise musical structures or implicit patterns in the raw notes. Though it may be relatively easy for a novice to pick up the basics of the tracking approach

and notation through tinkering with short phrases, the interface's limitations in catering for broader visual (or aural) overviews, and support for higher-level, macroscopic editing and arranging, make it difficult for inexperienced musicians to tackle longer, more complex musical forms, structures, and developments. Indeed, a similar trade-off is evident in other music programs founded on loops or short phrases (see sections 8.5 and 9.3), leading to a tendency to favour simpler musical forms, such as the progressive styles of dance music (house, trance, drum and bass).

A rift between direct and abstract musical control is even more pronounced in the sequencer, between performance capture and subsequent musical editing based on direct manipulation of musical structures. However, even during sequencer interaction, Collins (2005) found a “pre-occupation with small scale actions”. While he considers the consequent drop in productivity in contradiction to the composer's pursuit of a fast work rate (“achieving as much as possible”), this habit must be considered in the context of the user's subjective experience, where the narrowed focus leads to more flow-like interaction, like that described for tracking. Moreover, it is worth noting that, whereas productivity slows, such focused interaction typically corresponds to more rapid feedback, physical activity, and higher energy, which the user might perceive as a fast work rate, and find intrinsically rewarding.

Trackers provide an example of computer music interaction that bridges composition and performance practices; coupling more visceral, immersive, and engaging user experiences, based on reflexive actions and motor skill, with more abstract control of musical processes and time. Moreover, by identifying the mechanisms involved in supporting such interaction, program designers may be able to find ways to integrate them with more complex musical affordances.

8.3 direct manipulation for audio-based programs

The continuity of feedback is a central component of *direct manipulation* (Shneiderman, 1983), which leads UI designers to visual representations of music data that, while continuous, can be highly abstract. Rapid, shorter, more focused musical feedback, interleaves the domain representation with lower-level interaction (e.g. simple edits), towards not only support for more “continuous representation of objects of interest”, but through “rapid reversible incremental actions with immediate feedback” (see Table 2).

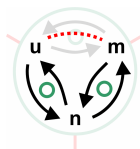
principles

- continuous representation of the objects and actions of interests;
- rapid reversible incremental actions with immediate feedback about the object of interest;
- physical actions and button pressing instead of issuing commands with complex syntax;

benefits

- helping beginners learn basic functionality rapidly;
- enabling experienced users to work rapidly on a wide range of tasks;
- allowing infrequent users to remember how to carry out operations over time;
- preventing the need for error messages, except very rarely;
- showing users immediately how their actions are furthering their goals;
- reducing users' experiences of anxiety;
- helping users gain confidence and mastery and feel in control;

Table 2 - The principles and benefits of direct manipulation,
as summarised by Sharp *et al* (2007), based on Shneiderman (1983).



measuring
directness

Sequencers and DAWs offer *graphical user interfaces (GUIs)* based on traditional applications of direct manipulation principles, developed for visual mediums. Digital tools and processes are linked to musical concepts through visual representations, which help trained musicians understand the workings of a program. Leman (2008) argues that this use of visual notations is harmful to music interaction, where users are involved in music only “indirectly”, through the visual proxy of notation (see also **system indirection**, Figure 4-9). While trackers do not abandon the advantages of a notation (e.g. abstraction as a tool), feedback is shifted to prioritise audio representations of the musical data, but in a way where the aforementioned goals and principles of direct manipulation are respected. This section has shown that this approach to *direct manipulation for audio*, using frequent, rapid, short episodes of audio feedback, confers the same advantages Shneiderman (1983) observed in visual mediums (see Table 2).

The previous section described the use of musical feedback to facilitate a user’s understanding and use of tracker notation. In this section, these analyses are extended to consider the editing activity and complexity of notation use that precipitates playback, and the effect of experience.

Figure 10 shows the elapsed time spent editing the music, between playback, for novice and expert tracker users, generalising trends observed in user sessions (see Figure 9 (a)). Logarithmic sampling is used, so that the area under the curve remains proportional to the number of episodes, using a log scale (see inset for an illustration of the intervals used). In the Expert case, the curve exhibits a log-normal distribution centred on a mode of 17.13s and median of 15.92s. For Novice users, the distribution is skewed towards considerably longer editing episodes, with a median of 67.15s and mode of 155.76s (2m36s).

However, a notable increase, relative to a log-normal distribution, is apparent at very short editing durations, below 10s (local minima, 1.30s), which may indicate inexperienced users tinkering with the tracker; making small changes to the notation, then using playback to understand their effect. This behaviour largely disappears with minimal experience.

The extent to which novice users are working slower, rather than simply longer, is indicated by Figure 11, which plots the number of edits (inputs that affect the data) between auditions, rather than absolute time. Here, experts, like novices, are shown to also favour individual edits, but as part of a wider trend towards shorter editing sequences (median = 2.36 edits), whilst novice interaction is still characterised by greater editing activity between requests for musical feedback (median = 5.44 edits).

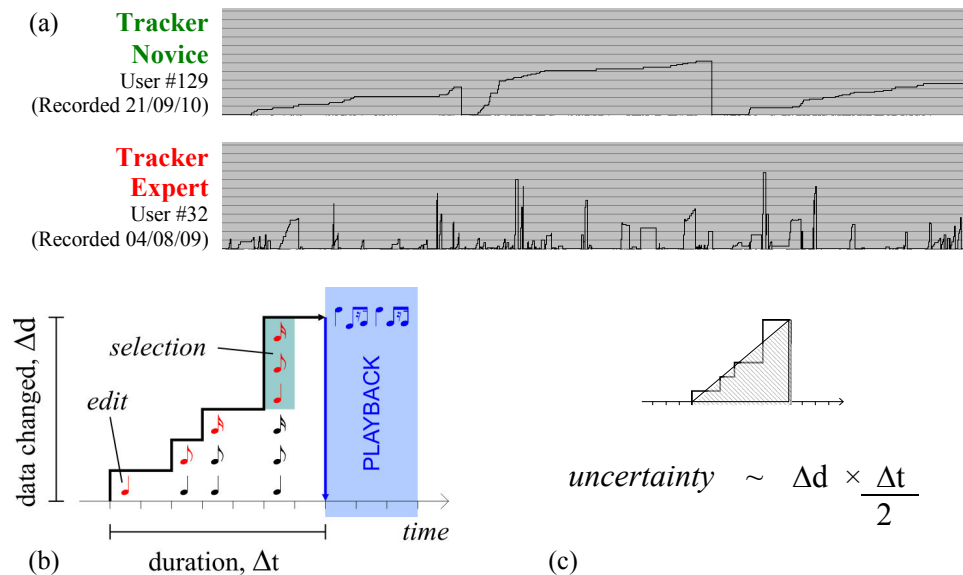


Figure 9 – Editing metrics and uncertainty. Examples, explanations, and definitions relating to analysis of editing episodes: (a) Total data changes plotted against session time, as visualised in *iMPULS|IVE* (see Section 5.4), taken from two representative session logs, with corresponding keyboard activity indicated on the time axis, in green; (b) Illustrative example of the roles of edits, selections and playback within an editing episode; (c) Proposed model for uncertainty, as used in Figure 12.

Table 3
Duration and editing statistics
from Figures 10-12

<i>editing episodes</i>		<i>novices</i>	<i>experts</i>
Duration	<i>median</i>	67.15s	13.24s
	<i>mode</i>	155.76s	17.13s
Number of edits	<i>median</i>	5.44	2.36
	<i>mode</i>	1	1
Data created/modified	<i>median</i>	5.70	4.00
	<i>mode</i>	1	1

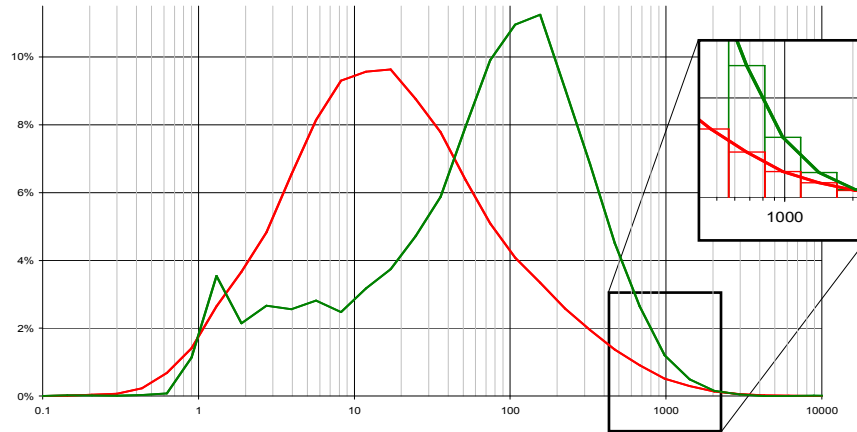


Figure 10 – Editing Episode Durations. The elapsed time (in seconds) between uses of playback, during which data is edited, for novice (green, n=548) and expert (red, n=574) tracker users. Data taken from sessions with over 30 minutes of interaction sampled logarithmically (see inset).

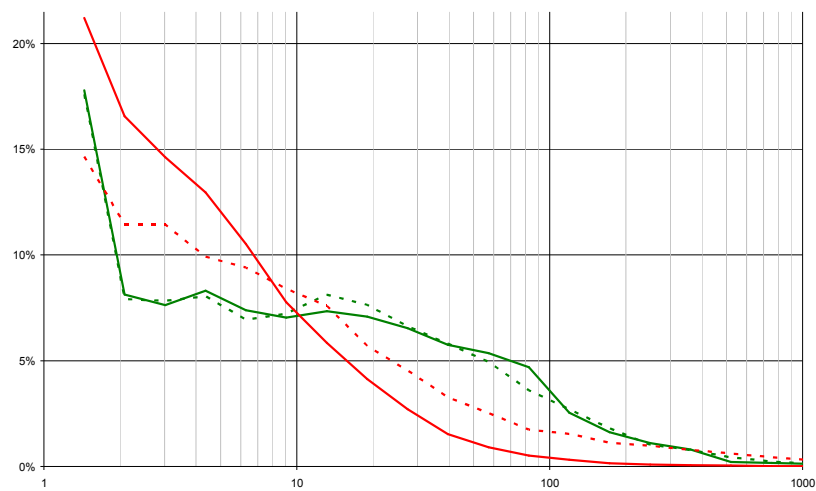


Figure 11 – Editing Activity between Auditions. The number of edit actions between uses of playback, during which data is edited, for novice (green, n=548) and expert (red, n=574) tracker users. Data taken from sessions with over 30 minutes of interaction sampled logarithmically. Adjusted (dotted) lines account for the increased scope of selections-based edits.

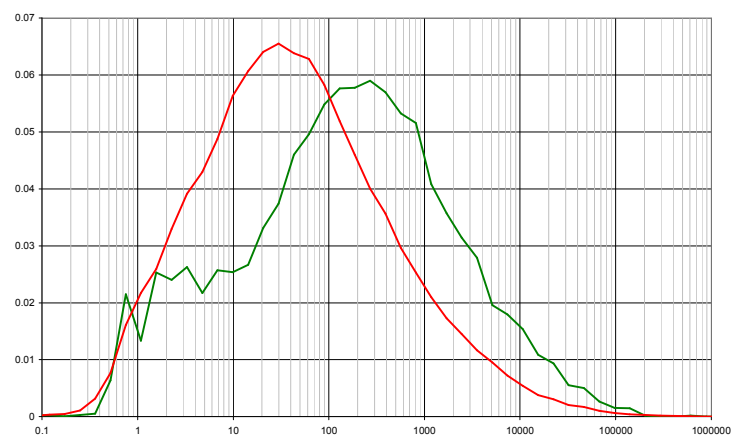


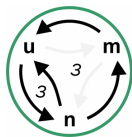
Figure 12 – Model and Plot of Editing Uncertainty. Uncertainty modelled as the product of data modifications and time between auditions (see inset equation), for novice (green, n=548) and expert (red, n=574) tracker users. Data taken from sessions with over 30 minutes of interaction (sampled logarithmically).

It is expected that expert users enter data more efficiently, exploiting features to manipulate blocks of music, rather than just individual notes. To account for the varying complexity of edits, logs were analysed to track the cumulative amount of data created (or modified) prior to each audition (illustrated in Figure 9 (b)), factoring for the use of selections and the clipboard. Figure 11 includes adjusted curves for both groups, which effectively weight edits according to the size of the selection they address. As the users' musical data is not collected, the data density of selections is estimated using a 25% coverage heuristic, based on tracker usage established from saved file summaries and publicly-available tracker songs.^{14,15}

The adjustment brings the expert profile closer to that of novices, who show only minimal selection-based editing (likely through drag-and-drop). However, the expert's greater tendency to make shorter edits remains, and thus neither speed nor efficiency can completely explain their more frequent use of auditions.

The variables modelled in Figure 11 and 12 can be seen as factors in the user's perception of *liveness*, which concerns the mapping of physical action to its effect on the domain (see 4.2.4). The difference between discrete levels of liveness is described by the nature and extent of the delay (in time or edits) in domain feedback, inherent in a notation or UI. Music software can exhibit Level 2 (manually-triggered), Level 3 (edit-triggered), or Level 4 (real-time) liveness. However, the findings above illustrate how experienced tracker users manually-trigger playback at or near the edit rate, influencing the effective liveness of the user experience.

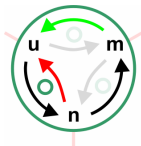
Greater liveness, through timely domain feedback, makes it easier for the users to understand the effect of their interactions within the notation, so far as they relate to the domain. As time passes, individual edits accumulate, making it harder for a user to recall and maintain a mental image of the product described, based solely on the abstract visual representation. Eventually, a *threshold* is reached, whereby it becomes necessary to execute (e.g. audition) the work to comprehend it. To account for the effect of both memory and editing complexity, Figure 9 (c) presents a hypothetical model of this threshold, modelled as uncertainty with



¹⁴ This corresponds to an average of one note every four tracker rows, or single musical beat. This fits with expectations regarding the use of the notation, balancing denser percussive patterns, which divide the beat, and sparser harmonies, which combine several beats (or bars) and tracks.

¹⁵ Note that a constant coefficient does not account for the tendencies of users, especially experts, to chunk selections into more abstract groupings (e.g. beat, bar, instrument, melody, harmony, pattern), which serve to make large selections more manageable. If we were to assume that most selections encapsulate single gestalts, such as a bar, then complexity is best modelled simply by the number of edits.

the notational representation that grows in proportion to both the number of changes in data, Δd , and passage of time, Δt (in seconds).¹⁶ Figure 12, as the product of the distributions in Figures 10 and 11, exhibits a log-normal distribution under the model, with modes of 225.02 for novices and 24.75 for experts.



Whilst greater expertise and literacy in most fields, including music, typically enables an individual to work more exclusively with written notation without recourse to live simulation, the findings in this section suggest this is not the case in tracking. In contrast, experience with trackers leads to a lower threshold, and sees tracker experts relying more on audio feedback (from the musical domain) rather than visual feedback (from the notation).

Unfortunately, the absence of equivalent data sets for other authoring software prevents the wider testing of the model, in respect of its applicability to more general interaction in music production and creativity. This work is left to future research. However, if verified, there are implications and applications of this metric, for UI design.

Less dependence on visual notation reduces the literacy threshold; tracker novices should instead seek to introduce more frequent auditions into their interaction. This frequency and the other editing metrics explored in this section (complexity, uncertainty), are measurements and calculations that can be made at the time of interaction. A program can track them and use them in support of the user's interaction and development. For example, music programs could display "liveness status",¹⁷ as a visual cue to encourage more frequent auditions. This might be as simple as a visible timer counting from the last instance of playback, a counter of the un-auditioned edits, or varied shading of data to indicate when, if ever, it was last heard. More complex implementations might track a user's average over time, activating a response only if it deviates sufficiently from an established optimum.¹⁸

In the samples studied, novices have tended towards longer editing episodes. Some evidence has been identified for the use of very short edits, possibly as a learning device. In Figures 10 and 12, these appear as local maxima, outside the main log-normal

¹⁶ This variable can be seen to approximate the area under the line, in Figure 9 (c).

¹⁷ For example, as a timer counting from the last instance of playback, or a meter counting the subsequent number or extent of edits. Sections of music could also be colour-coded (e.g. desaturated) to reflect the times they were last heard live, creating a heat map signifying editing activity within a piece. Rothermel *et al* (2000) have used similar visual feedback mechanisms to indicate "testedness" in spreadsheet use and software development, representing the degree to which a formula or block of code has been executed (tested). Notably, however, the objective in artistic creativity is to support a user's focus on their music, rather to encourage "correctness" or guard against "overconfidence" (see Rothermel *et al*, 2000).

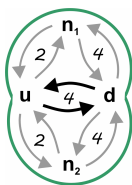
¹⁸ For example, guiding the display of tips or advice (see *Future Work*, Section 10.4).

distributions, this time below the averages for experts. In these cases, extensive use of playback is punctuated by trivial edits that limit a user's productivity, if maintained. This bimodal distribution may suggest a *goldilocks principle*; a happy medium between editing that is too long or short, too complex or trivial, or involves too much or too little uncertainty. In this way, an individual's use of playback in managing liveness may be related to flow theory's *balance of challenge and ability* (see 3.7), and the "flow channel" that lies between boredom and anxiety (Figure 3-7). The challenge of mentally simulating a visual representation of the music is mitigated by aural feedback, but greater ability to work without this scaffold benefits overall productivity and allows the individual to tackle greater musical challenges.

8.4 visual feedback and window use

To support the "many paths" taken by creative individuals, music programs provide extensive and varied tools for visualising and manipulating data. Though command syntaxes and keyboard shortcuts require no visual cues, popular usability techniques (e.g. Nielsen, 1993) encourage the use of visual metaphor to advertise and explain program functionality, notably through the use of icons. Consequently, limited screen space requires more complex authoring programs to distribute program functionality over multiple windows, which not only hide aspects of the interface, notation and data, but require the user to manage their creative environment – finding, switching, and arranging views and window, before data can be manipulated.

*focusing
and learning*



This section looks at the use of windows in music software, and their impact on the creative user experience. Windows are the containers for visual feedback, and can greatly influence the user's ability to maintain the focus and concentration necessary for flow. Multiple views (or multiple *notations*, as in Figure 4-10a) can provide a broader and flexible perspective of a project, but also divide the user's attention. Moreover, segregating aspects of the notation can introduce *hidden dependencies*, and varying visual formats (especially of similar data) can reduce *consistency*, making it harder for the user to get a grasp of the program or their data. By comparison, more unified views support less flexibility, but direct and maintain the user's focus in one place. In combination with fixed layouts, the program also becomes easier to memorise and anticipate, requiring less management.

Figure 13 identifies styles of window layouts commonly seen in music programs. Using visual metaphors to separate hardware devices in the electronic studio (Duignan *et al*, 2004), sequencers and DAWs naturally lend themselves to distributed window layouts in digital music production – mixers, transports, effects, samplers, synthesizers, tape decks. However, the WIMP approach also imposes the “desktop metaphor” on this virtual studio, in which each device is now also a metaphorical “page” on the virtual desktop. Banks of dedicated, physical buttons, devices and displays are replaced by flat, abstract, and mutable virtual representations, contending for space on a confined screen. In most cases, the metaphor is stretched to enable users to resize, scroll and zoom, as well as simply move and arrange, different views.

Figure 14 shows how the design of window layouts in music programs translates to the observed number of concurrently active windows in the user interface, each of which must be managed by the user or program. Figure 15 plots the number of windows against the corresponding frequency of user inputs devoted to managing them, where a positive correlation between the number of windows and the level of management is evident. The relationship follows a power curve ($R^2=.525$), with minimal window management for single-windowed programs (with some popup dialogs), significantly increasing for multi-windowed workspaces, but which are then also able to accommodate more windows at limited extra cost. As such, the underlying window style of a program (see Figure 14) seems the strongest predictor of both window numbers and management, despite variations in design and use. In Figure 15, clusters are drawn to delineate both single-windowed approaches and those supporting more flexibility in moving, sizing, or layering (i.e. MDI and floating windows).

Expectedly, simple dedicated plugin hosts (shells that connect plugins to audio hardware, but offer little or no editing functionality of their own – e.g. *SAVIHost* and *MiniHost*) employ the fewest windows, simply providing a frame for the plugin (i.e. *reViSiT*).¹⁹ A plugin itself, *reViSiT*’s tabbed-window design and use of the keyboard to expose program functionality also enable minimum window use.²⁰

¹⁹ *VSTHost*, another dedicated plugin host, is more advanced, supporting multiple plugins, with multiple windows per plugin (e.g. settings and plugin UI). *energyXT* is also used in a similar capacity.

²⁰ To gain absolute control over keyboard input, *reViSiT*’s UI must be separate from the host frame. However, the VST architecture requires a plugin to specify a UI for display by the host. For this purpose, *reViSiT* provides a simple, minimal toolbar, styled as a transport that merely duplicates selected functionality available in the other window, and is expressly designed to be ignored by the user.

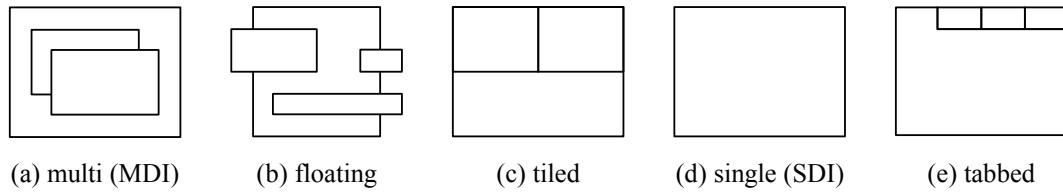


Figure 13 – Archetypal window styles used in music software. Programs often employ more than one style (e.g. tabbed floating windows; MDI client windows and floating toolbars), as shown in Figure 14.

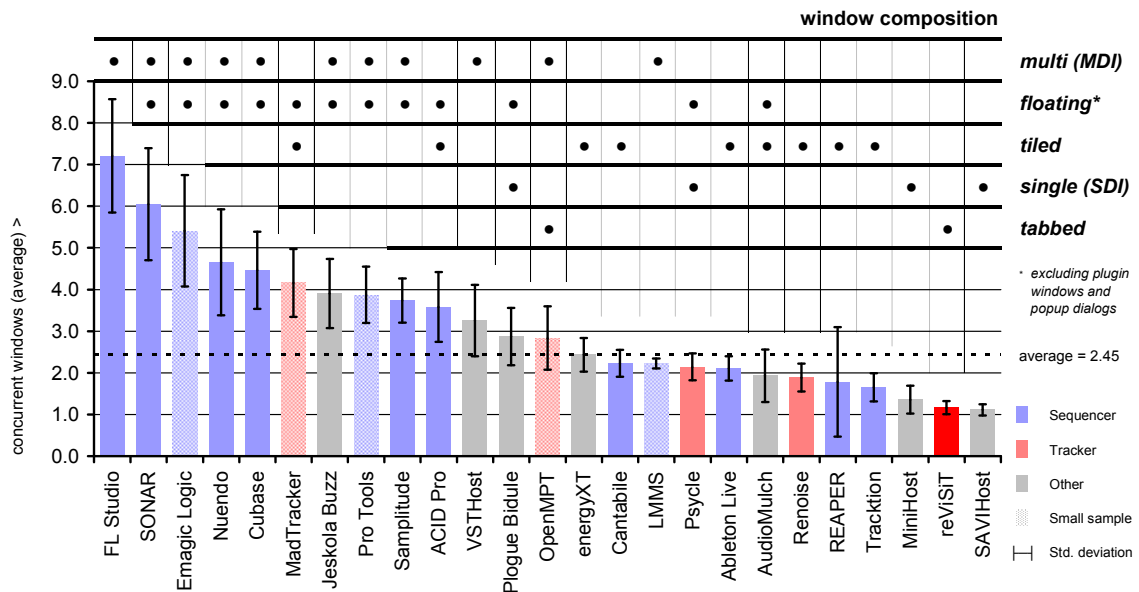


Figure 14 – Window layouts by program. Plots window layout styles and average number of concurrent window in music programs, colour-coded by family. Figures exclude non-interactive static windows, such as MDI frames and reViSiT toolbar. (Small samples, based on less than 5 users, are also indicated.)

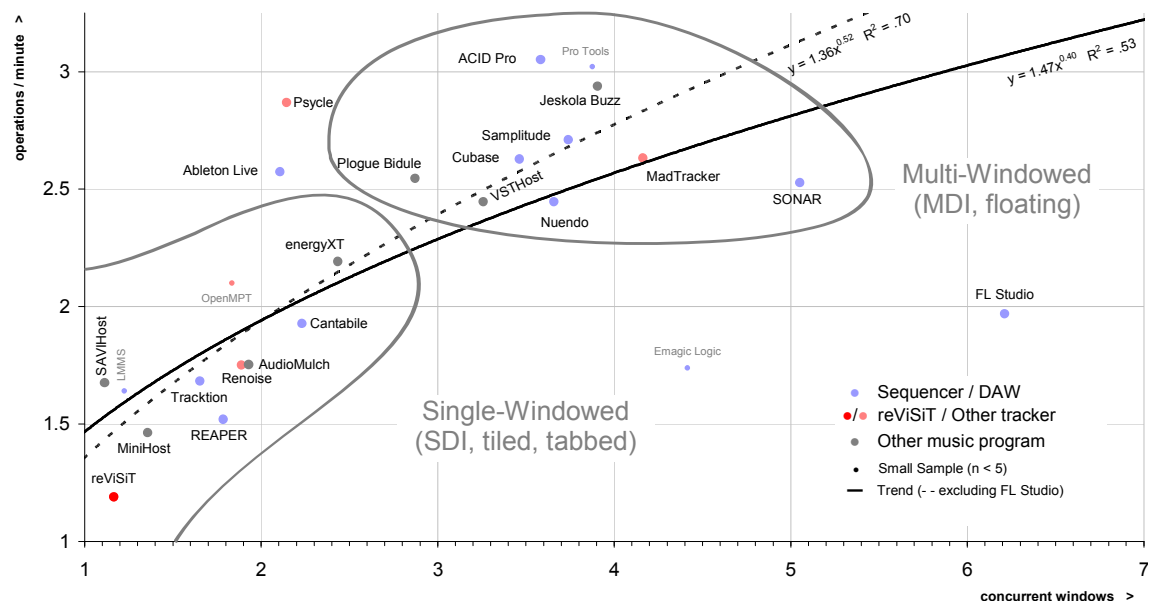
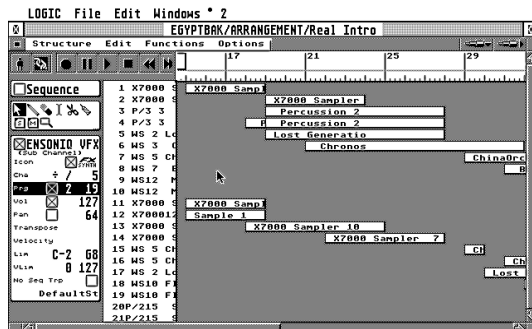


Figure 15 – Window management in music programs. The amount of window management (operations per minute, including switching, showing, hiding, moving, and sizing), plotted against the average number of concurrent windows. Two trend lines are plotted, including and excluding the *FL Studio* outlier (see Section 8.5). Clusters delineating single and multi-windowed UIs are also shown.

the sequencer legacy

Four of the five programs with the most prolific use of windows represent long established platforms for music production – *Cakewalk SONAR*, *Emagic Logic*²¹, and *Steinberg Nuendo* and *Cubase*. These programs average between 4.5 and 6.0 concurrently active windows, but use varies considerably between users (standard deviation ~ 1.5), indicating the increased flexibility, but also management requirement. The legacy of these programs, which each date back to the late 1980's and the original rise of GUIs and desktop metaphors, is still clearly evident in the window styles they employ (see Figure 16 (a) and (b)). Though aesthetics have improved over the years, the conservative use of generic window styles provided by OS APIs (*MFC*, *WPF*, *Carbon*, *Cocoa*) remains evident. However, SONAR's uni-platform approach enables it to avail of platform specific window mechanisms to deliver a greater number of windows more efficiently. Though APIs progress, manufacturers looking to support multiple platforms (including older OS's) are restricted to functionality common to each, limiting the adoption of newer methods, and restricting programs to a lowest common denominator subset of features.

Figure 16
Screenshots of
sequencer and
DAW interfaces.



(a) *Logic 1.5* (C-Lab, 1993)



(b) *Logic 8* (Apple, 2008)



(c) *Live* (Ableton, 2006)



(d) *Cubase 4* (Steinberg, 2006)

²¹ Logic has changed ownership several times, most recently to *Apple*, who withdrew support for Windows users. Data in this study thus comes from older versions of the program (v5.5), by *Emagic*.

Whilst multi-window styles coped with the simpler functionality of early MIDI sequencers, the greatly extended capabilities and audio production remit of the modern DAW lead not only to an unmanageable number of floating windows, but an increasingly distributed sense of UI focus. For example, Cubase's main *project window* (arrange view) appears as a generic child window, beside more peripheral parts of the UI (e.g. time display, settings window). Indeed, unlike other parts of the notation, this central view seemingly has no dedicated menu or keyboard shortcut.

housekeeping tasks

That these mechanisms are not always apt is highlighted by Collins' (2005) observation that sequencer users must frequently interrupt their work and perform "housekeeping tasks" to keep the program manageable. In this research, logs show that users often simply ignore this requirement, labouring on in cluttered workspaces (e.g. Figure 6-4 and Figure 16 (d)) layered with unclosed yet unused windows, despite unused space around them, into which windows could be moved or expanded, but never are.²² In a well-defined production environment built on standard processes, users can benefit from optimising their workspace for common tasks; but, in fast-paced, spontaneous, and unstructured creativity, tasks are more varied, planning ahead is harder, and time is more precious. Unless the program becomes unusable, artists are reluctant to invest their attention in housekeeping activities, the return on which is too abstract or distant for them to appreciate.

*case study:
FL Studio*

The heaviest use of windows is seen in *FL Studio*, which fills the workspace with numerous smaller views and devices. Contrary to expectations and the trend in Figure 15, the program exhibits good performance with regard to window management and flow (see Section 9.3). When *FL Studio* is treated as an outlier and omitted from the regression analysis,²³ the fit of the resulting model dramatically improves ($R^2=.704$), as shown in Figure 15. Section 8.5 presents a brief case study of the program to explore this anomaly in the context of other aspects of interaction – noting that, while windowing does present focus issues, the overall impact on the user experience is partly mitigated by specialist UI provisions, and the availability of musical, rather than visual, feedback.

²² An effort to accurately gauge how program views contend for screen space was also made, by summing the total area of active windows and dividing by the visible area of the workspace. This *contention ratio* accounts for the relative size of windows, so that programs aren't penalised for smaller windows that obscure the background less. Results ranked programs similar to that in Figure 14, with multi-windowed environments hiding up to 60% of the active workspace. However, the metric showed limited predictive power when correlated with window-related activity ($R^2=.44$), cognitive dimensions, or flow. The lack of relationships between this metric and other interaction properties would also seem to confirm users' disregard for maintaining and optimising their workspaces.

²³ *FL Studio* lies over 3 standard deviations below the values predicted by either curve, in Figure 15.

More recently-introduced music programs, such as *Ableton Live* (2001-), *Mackie Tracktion* (2003-) and *REAPER* (2007-), adopt integrated, tiled approaches that unify core functionality into a single, main screen, as indicated by the significantly lower proliferation of windows in Figures 14 and 15.²⁴ Trackers, such as *Renoise*, *reViSiT*, and *Psycle* generally also focus on a central, main view of the tracker notation, though others, such as *MadTracker* and *OpenMPT*, have tried to marry older DOS layouts with more standard Windows™ GUI methods, incorporating floating tool windows and MDI clients reminiscent of sequencers.

Ableton Live, however, receives almost no benefit from its minimal window layout. Apart from preferences dialogs, *Live*'s own UI is careful to minimise the use of windows. Instead, the problem can be attributed to floating plugin windows, which the full-screen, single-windowed application is less well equipped to accommodate, creating extra work for the user. The UI supports only limited windows concurrently, but many different (possibly large) windows are used during the course of interaction, requiring frequent opening and closing, hiding and showing – windows are distributed over time, rather than space.

Support for plugins plays a significant factor in the proliferation of windows in music authoring environments. A flexible plugin architecture gives third-party developers a free hand to add functionality to a program that was not in the original design. This foreign influence on the user experience introduces UIs that vary in layout, size and interaction styles, and cannot easily be accounted for or seamlessly integrated into the host's main UI. In music programs, plugin effects and synthesizers (including *reViSiT* itself) thus appear as separate windows, either within an MDI frame or floating above and outside the host UI. Moreover, with one or more plugins often assigned to individual tracks, a single project can host dozens of plugins. The resulting number and variety of UIs, aesthetics, and interaction styles adds clutter to the workspace, reduces the consistency and cohesion of the overall user experience, harming the user's *sense of control*. While the architects of plugin specifications should perhaps look for ways to encourage greater uniformity in plugin UIs, host manufacturers should look to more adaptive, integrated approaches for hosting and managing plugin windows.

²⁴ Aspects of these streamlined workspaces can also be seen in the recently-announced *Cakewalk SONAR XI* and *Apple Logic Pro 9*.

Figure 17
Window metrics vs. cognitive dimensions. Window metrics taken from user logs, plotted against survey results from the 7 most-used programs in the study.

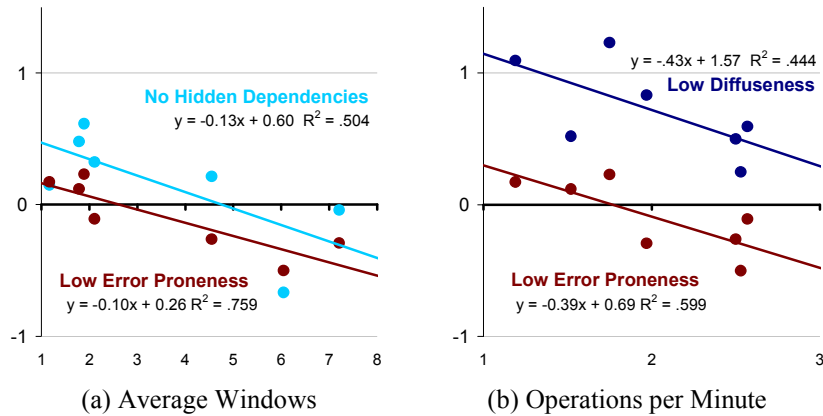


Figure 17 draws on user surveys, covered in the next section, to highlight the impact of windows and window management on the cognitive dimensions of a notation, based on the seven most frequently used programs in the study (*reViSiT*, *REAPER*, *Cubase/Nuendo*, *Ableton Live*, *FL Studio*, *Renoise*, and *SONAR*). Greater numbers of windows engender *hidden dependencies* ($R^2=.504$) and increase the user's *error proneness* ($R^2=.759$), which are harmful to a user's *sense of control*, make it harder to maintain *concentration & focus*. The knock-on requirement to manage greater numbers of windows likewise correlates with *error proneness* ($R^2=.599$), but is also closely associated with a user's perception that a program is too diffuse, making worse use of space (*diffuseness*, $R^2=.444$). Other aspects of cognitive dimensions and flow are further explored in the next section.

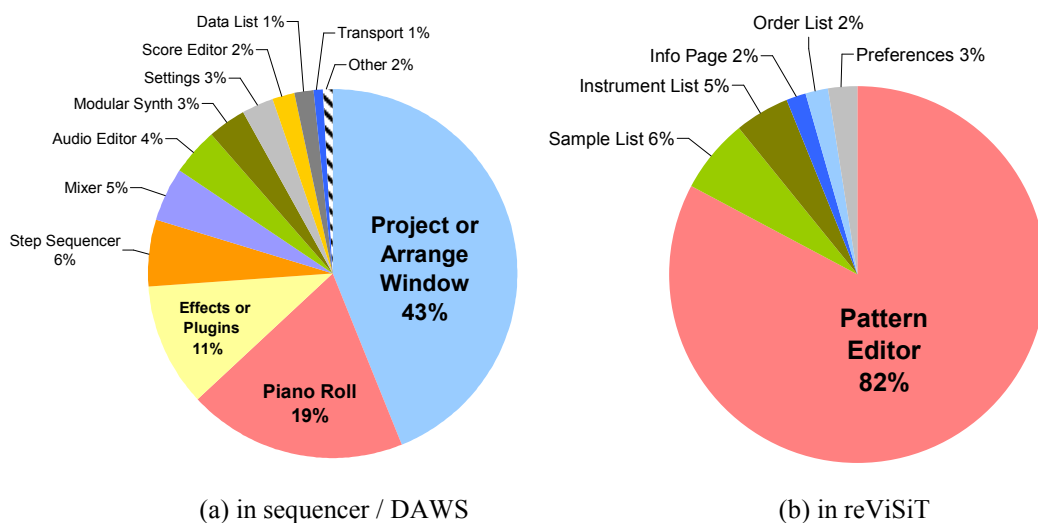
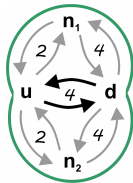
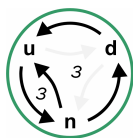


Figure 18 – Users' main focus in music software. Percentage of users with given main focus (a) in sequencers, based on survey response (n=191) and (b) in reViSiT, based on log data (n=1125).

dispersed focus
in sequencers



pattern editor
as focal point



window use
in context

The precise roles of individual windows within the various hosts are difficult to extract from raw window data in the logs, but responses from the end-of-experiment survey shed some light on sequencer use. Figure 18 (a) shows which device or component part of a program is seen as the focus of a sequencer user's interaction, out of the various standard views offered in modern DAW software. The *arrange view* (or *project window*) stands out as the most common focus in the sequencer (43% of users), offering macroscopic view of the tracks within a piece, plus facilities to arrange parts and record performances using MIDI or microphone input. Users who rely less on recording turn to computer-based note entry and editing using the *piano roll* (19%) – especially in *FL Studio*, where it is focal to over half (55%) the users. Similar editing functionality in *score editor* views is much less central to the sequencer user experience (2%), despite a relatively high literacy rate among users (46%). Lastly, a fifth (20%) of users spend the majority of their program time in parts of the sequencer that offer no direct editing facility (mixer, effects, plugins, settings, transport bar) – presumably concentrating their creativity on interaction with external audio and music hardware, using only the recording, synthesis and DSP processing offered by the software.

By comparison, data from *reViSiT* interaction (Figure 18 (b)) showed a central and fundamental role for the Pattern Editor, the interaction focus for 82% of users. Most like the piano roll, this screen governs note-level music editing, but it also enables a degree of low-level arrangement, through block selection and clipboard operations (see section 7.2), and offers a wider scope, across all tracks, parts and instruments at any one moment in the piece. Though trackers cater less well for more macroscopic editing and overviews of the music, offered by a sequencer's *arrange window*, the user's attention and activity is significantly less divided between diverse and separate sub-devices, leading to a more concise perception (lower *diffuseness*) of the notation, and defining a clear visual locus for the user's *concentration and focus*, also indicated in the survey detailed in Chapter 9.

Ultimately, while there is little doubt that greater numbers of windows lead to more complicated screen layouts and more frequent housekeeping, often at odds with exploratory creativity, their ultimate impact on the user experience depends on context, and the provisions for window management within any given program. In respect of virtuosity, consistency and predictability are key to enabling the learning and fluid execution of editing activities in any creative environment, yet are defeated by dynamically

changing and reconfigurable screen layouts. Moreover, if music programs are to support spontaneity and exploratory creativity, forcing the user to plan and manage the creative environment only detracts from interacting with the music.

Interaction and window management becomes most complicated when programs mix UI styles, especially when free-floating windows are supported both inside and outside a program's main UI, as in the combination of MDI containers and floating, always-on-top toolbars or other views in many DAWs. *Ableton Live* also highlights the problems when only a few floating windows, such as those belonging to plugins, are introduced to a host environment that is not designed to accommodate them. To conclude this review of window use in music programs, the next section presents a case study of *FL Studio* and its prolific, yet efficient use of windows, in what at first sight appears to be an exception to the rule, but ultimately exemplifies many of the findings discussed in this and previous sections.

8.5 case study: *Image-line FL Studio*

FL Studio (Figure 19) is a WIMP-based DAW, which draws heavily on visual metaphors to the electronic studio. This thesis has highlighted issues with similar approaches to music software design, as concerns the reliance on the mouse, visual metaphor and windowed environments. This case study briefly looks at how these potential problems are handled in *FL Studio* – which, despite exhibiting the highest number of concurrently active windows recorded in our study (see Figure 14), nonetheless shows evidence of supporting flow experiences.

Figure 19
Screenshot
of *FL Studio*.



In his review (Harding, 2010), sound engineer, Julian Harding, states:

FL Studio is still my 'turn-to' sequencer for anything needing sampling, MIDI sequencing or virtual instruments, and it's much quicker for turning an idea to reality than any other environment... it is worth becoming familiar with the 'individual' sequencing methodology in FL Studio for the sheer speed and satisfaction available... no DAW makes work quite so much fun as FL Studio.

In the end-of-experiment questionnaire (see section 9.3), *FL Studio* users (n=29) similarly scored the program highly with regard to *direct & immediate feedback* (+0.80), *loss of self-consciousness* (+0.82), and *intrinsic reward* (+1.18). The overall flow metric of +0.73 was likewise high, relative to other sequencers. Notably, *Ableton Live* exhibited a similar profile, but only surpassed *FL Studio* with respect to *concentration & focus* – the integrated UI of *Live* (+0.75), and those of trackers (+1.00), score higher than *FL Studio* (+0.59), which more closely mirrors the performance of other windowed sequencers (+0.56). Moreover, despite a generally strong cognitive dimension profile, windowing may also explain low scores for *hidden dependencies* (-0.04), which also give rise to *error proneness* (-0.29) and *hard mental operations* (-0.38). For example, it can be difficult to track the flow of an audio signal between devices (windows) in the program.²⁵

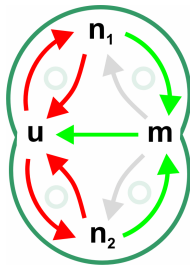
*a pattern-based
step-sequencer*

The program began as a simple emulation of a hardware step-sequencer, but later expanded to cover the full remit of the studio, by mimicking the appearance and function of other hardware devices. As in the tracker, musical time is represented by a grid, typically representing one bar divided into 16 quarter-beats, where a song is formed by sequences and layers of grids, called patterns. Unlike the text of a tracker, each cell in the pattern contains a toggle switch that is linked to some form of one-shot sound source, such as a sample or MIDI part.

*fast, frequent
musical feedback*

The limited length of step sequencer patterns best suits drum programming and simple melodies based on repeated loops and variations of 1-bar phrases, as found in popular, “four-on-the-floor” dance music. Editing is normally conducted with the pattern playing on repeat and the user toggling 16th-notes in the grid to

²⁵ Some programs offer more explicit representation of signal flow. Modular programs (*Buzz*, *energyXT*) use a visual programming language style, drawing lines between objects of interest. *Reason* more literally displays the wires between devices. Other programs, such as *Tracktion* or *Live*, enforce a left-to-right or top-to-bottom flow of audio signal, where audio channels become ordered lists of devices. See (Duignan, 2007) for a detailed analysis of different approaches.



integrated creative environment

create and edit the music, during playback, dedicated control of which is exposed using the play, stop, and skip keys on multimedia keyboards. This constitutes a very ‘live’ creative environment, where edits are quickly realised, at the next iteration of looped playback, but do not require the constant input or skill of a realtime musical performance (see Section 4.2.4). Instead, users tinker with the on/off states of the grid, at their own pace, until they arrive at a pattern they like, which can then be committed to the song. Combined with a large library of preset sounds, this engenders a very low learning threshold, where novices can quickly produce ‘professional sounding’ music, albeit in a limited range of styles. With experience, more complexity and creativity is possible through the piano roll, and the editing of presets and sound sources.

In contrast to other DAWs, *FL Studio* is designed for creating and editing material *within* the program – using virtual instruments and devices, rather than as an adjunct to a recording process supported by external instruments or hardware. The user’s focus remains within the program for the duration of the creative process, albeit divided across its constituent components and windows. By restricting windows to the confines of the MIDI client, the user’s area of activity and awareness are unified, facilitating *action-awareness merging*. Control is also highly optimised for the mouse and handful of keyboard shortcuts, obviating the need to switch between input modalities and devices (e.g. acoustic, audio, MIDI).

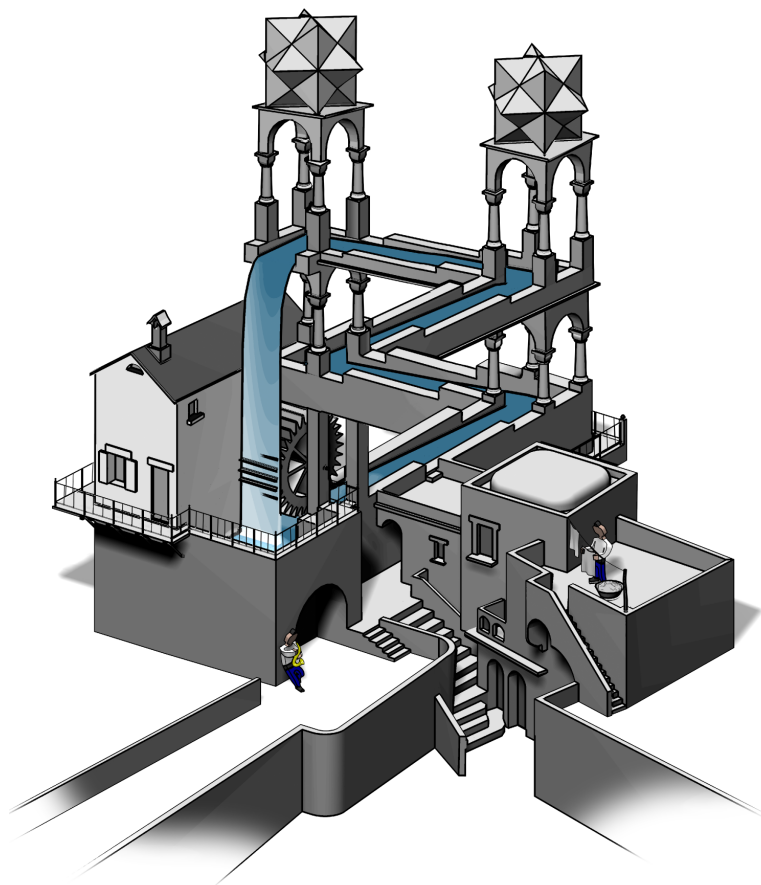
The high number of windows is also, in part, mitigated by the simple functionality of the devices themselves, which represent basic, generic simulations of their hardware counterparts, in contrast to more literalistic, faithful reproductions of specific hardware marques. As an integrated suite of tools, devices can be designed to complement each other while also applying consistent, unified UI techniques, which allow users to easily switch between editing contexts. By comparison, working environments built on diverse, independently-developed machines (e.g. external devices, plugins) allow more flexibility and depth, at the risk of duplicating and dispersing functionality, compartmentalising the creative process, and lowering *consistency* (of aesthetic, interaction styles, etc.) between different parts of the UI.

streamlined window management

Most windows are of fixed size and layout, facilitating learning and organisation, and expose select windowing functions as mouse gestures, which vary depending on the window’s role: while a double-click serves to maximise the main track view, it will “roll-up” windows less central to editing. Moreover, while sizable windows can be scaled with the conventional, yet fiddly drag of the

thin outside frame, the environment also allows their height to be more quickly, if less directly, manipulated by right-clicking the caption and moving the mouse up or down. Such provisions streamline housekeeping tasks within *FL Studio*; explicitly catering for more advanced and experienced users, with methods that trade the consistency and visibility of direct-manipulation, for quicker, non-visual, well-learned behaviours. Because of this, and because *FL Studio* focuses more editing activity within single windows, the corresponding rate of windowing activity (1.97 ± 0.24 commands/min) is comparatively low, given the large number of active windows (7.85 ± 0.47) and high contention ratio (2.52 ± 0.12).

Harding's review relates the speed of his creative process with the fun and reward involved; describing use of the program as one might a sketching process, taking musical ideas down as quickly as possible. *FL Studio* supports a live editing environment, partly enabled by narrowing the focus of musical edits. In the same review, Harding admits to having to resort to other programs for more complex aspects of musical production. Perhaps significantly, of all the sequencers tested, *FL Studio* was seen by its users as the easiest to master (*virtuosity* = +1.25), which may indicate that, while the program supports a low entry threshold through the provision of simple primitives, the ceiling of musical complexity is also lower than other programs.



chapter nine **flow and cognitive dimensions**

Towards the end of the experiment and after registration had closed, a second questionnaire was issued to users to gauge their subjective experience of *reViSiT* and tracker notation, notably in comparison to their experiences with a specific sequencer with which they were familiar (e.g. the host sequencer), with respect to flow and the cognitive dimensions of notations framework (Green, 1989). Participation in the questionnaire was incentivised by the offer of a new version of *reViSiT Pro*, which in addition to a handful of new features (in part based on lessons learnt during the research¹), also removed the interaction logging component.

Questions were presented in three parts, for both *reViSiT* and their chosen program. The first is based on interaction styles and preferences, repeating questions in the initial questionnaire (see Section 5.3.1 and Appendix C), this time focusing exclusively on the user's experience of the program in question. In sequencer survey, these were supplemented with questions to probe details of sequencer use beyond that available from the collected interaction data.

¹ Support for MIDI files (*open interchange*) and pattern annotations (*secondary notation*).

The second part of the questionnaire presented two blocks of statements describing the 9 components of flow, which the user was instructed to score on a 5-point Likert agree-disagree scale, with respect to how they perceived them in the user experience. This section and the flow statements were adapted from the *Dispositional Flow Scale-2 (DFS-2)*, developed as a psychometric test to quantitatively measure flow in a given activity (Jackson and Eklund, 2002), which has also been successfully applied to HCI contexts (Wang *et al.*, 2009). In contrast to other applications of the technique, each flow component was probed using only two statements (rather than four) to reduce the overall length of the questionnaire, which might otherwise deter participants.²

The third and final part uses similar Likert scales to score a single block of sixteen statements corresponding to cognitive dimensions of the notation, which enables comparisons and correlations to be made between flow components and properties of the notation. Statements were based on the *Cognitive Dimensions Questionnaire Optimised for Users*, which presents each dimension in language that could be interpreted by end-users.

An additional *virtuosity* dimension is introduced in an effort to assess ‘learnability’ properties of a notation, not captured by the original framework (Elliot, et al., 2002). Here, it is tested using the statement “With time, I think I could become a virtuoso user of the system”, corresponding to how easy a user believes a notation is to master, in line with flow’s *balance of challenge and ability*.

Questions were presented twice, once for *reViSiT* and once for the user’s chosen sequencer, which they selected from a preset list of 12 common packages or specified themselves.³ Users also had to state their level of experience with the program and select, from a list, which device or component part of the sequencer they spent most time in (Arrange window, score editor, etc.).

² Consequentially, though answers can be averaged to produce a more accurate response (less subject to the wording of flow descriptions), reliability statistics are not calculable. However, previous use of the technique in an HCI context (Wang *et al.*, 2009) testifies to the soundness of the method.

³ Despite explicit instructions to the contrary, a number of users answered this section about another tracker – possibly heralding from *Renoise*’s marketing of the tracker as a “DAW”, but possibly due to the respondent’s lack of knowledge concerning more traditional sequencers. This oversight, however, proved useful in providing an alternative source of impressions on tracking (see text).

9.1 interaction styles and preferences

Table 1 shows the results of the survey with respect to interaction styles and preferences in (a) *reViSiT* and (b) the user's chosen sequencer or DAW. A total of 254 completed questionnaires were received, with 9 excluded as invalid (offering the same response to all questions). Of the 245 remaining responses, 191 (78%) describe sequencer-style DAWs, 14 (6%) describe other trackers (e.g. *Renoise*), and another 40 (16%) describe other types of music program, such as score editors (e.g. *Sibelius*), audio tools (e.g. *Audacity*) or modular graph-based tools (e.g. *energyXT*, *Buzz*).

input device
preference

Despite the keyboard emphasis in the tracker UI, equal numbers of *reViSiT* users spend considerable time on both the keyboard (42%) and mouse (43%), possibly as a result of the plugin's integration with mouse-oriented hosts. With experience, however, users move away from the mouse ($r=.33$), towards a preference for the keyboard ($r=.29$). Experience with other (standalone) trackers also correlates with an avoidance of mouse-based ($r=-.20$) and MIDI ($r=-.22$) input modes. Indeed, a negative correlation between a preference for the mouse and its actual use ($r=-.20$) may reflect the relative lack of mouse support in the UI's design.

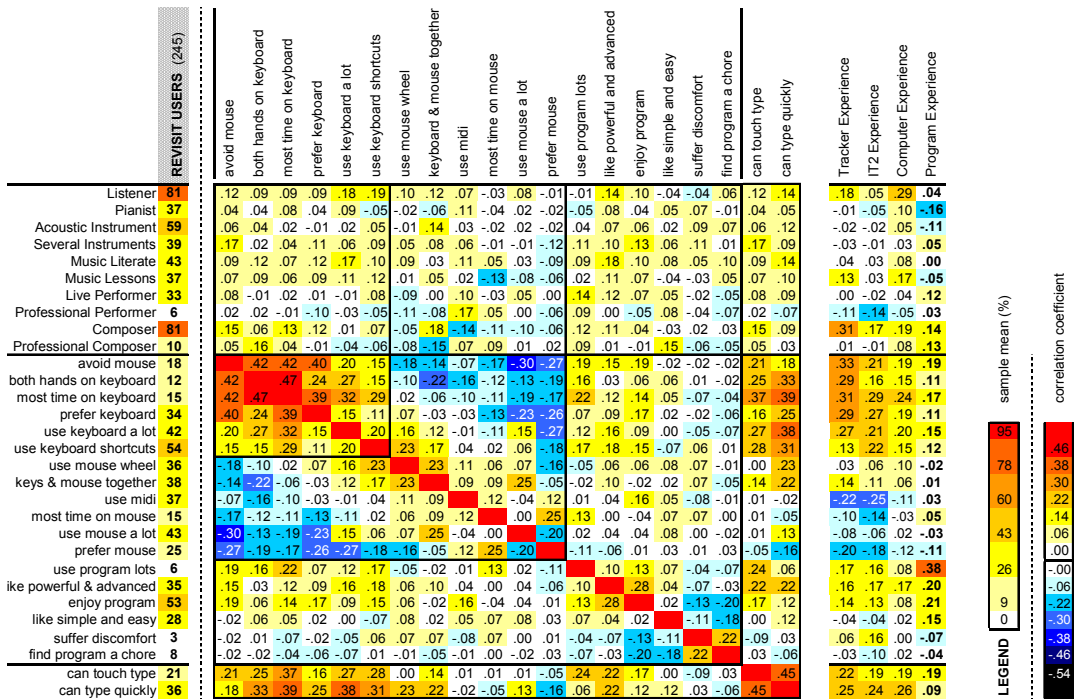
enjoyment and
hardware use

53% of participants enjoyed using *reViSiT*, compared to 73% for other software, typically their preferred music program. Remarkably, SONAR users – which combines extensive use of the keyboard (42%), mouse (67%), and MIDI (50%) – appear to universally enjoy their program (100%), though the sample is both of limited size (12 users) and contains no professional users. *Ableton Live* (87%) and *REAPER* (63%) also stand out as enjoyable programs. *Live* users, notably, appear to rely heavily on hardware control, through MIDI controllers (87%) and control surfaces (54%), as well as keyboard shortcuts (64%) – all of which facilitate fast and direct control of the program. Use of MIDI is similarly common in *REAPER* users (71%).⁴ Indeed, only 12% of sequencer users stated that they didn't use any specialist hardware.

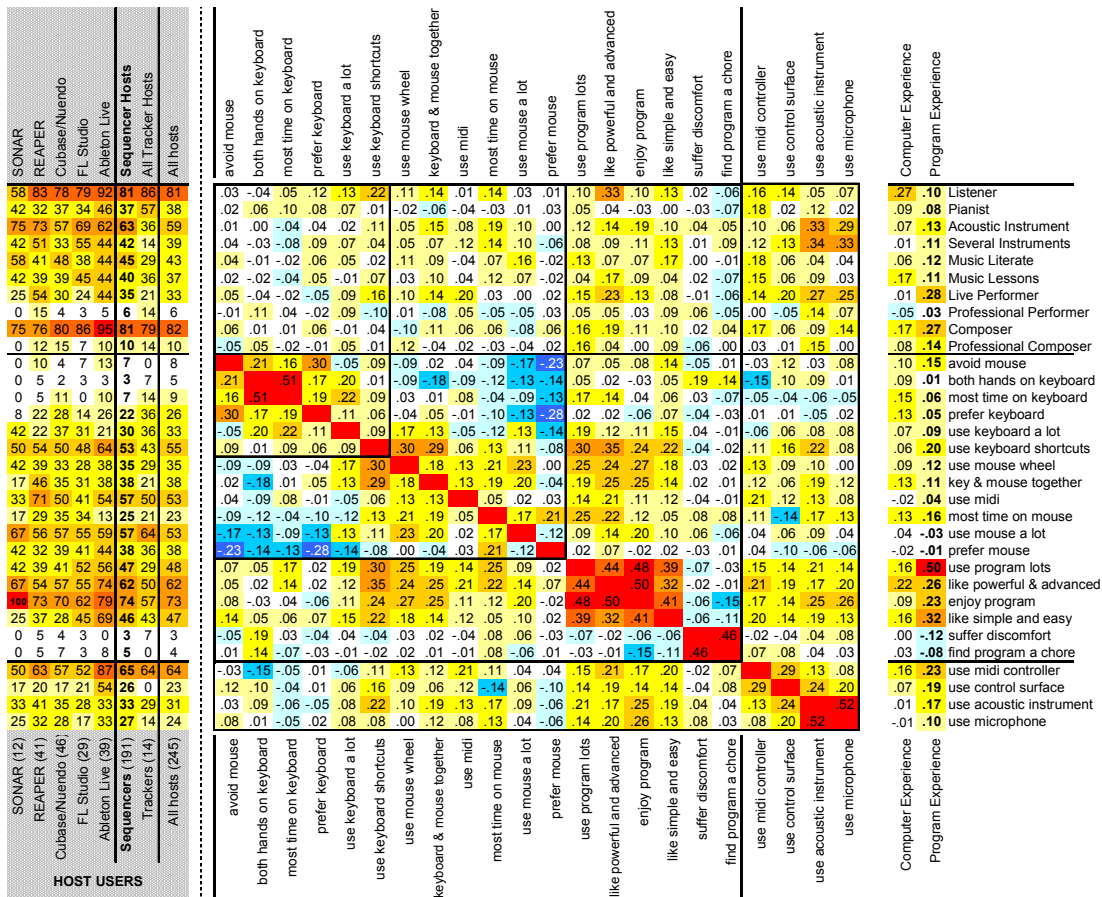
Sequencer users also saw little contradiction between programs that are “simple and easy”, and those “powerful and advanced” ($r=.32$), both showing moderate correlations with enjoyment ($r=.41$ and $r=.50$), which may reflect the mixture of usability techniques and advanced functionality in modern DAWs. By contrast, *reViSiT* users' enjoyment correlates significantly more with greater functionality ($r=.28$), compared to usability ($r=.02$) ($p < .05$).⁵

⁴ Though not evident in the survey data, *REAPER*'s results might also be boosted by the demographic of its audience, which notably attracts amateurs and hobbyists, for which enjoyment is key.

⁵ Tested using *Williams' T2* statistic ($r_{12}=.27$, $r_{13}=.02$, $r_{23}=.04$; $t=-.3.025$; $df=242$).



(a) reViSiT Questionnaires



(b) Sequencer / DAW Questionnaires

Table 1 – Music Experience & Interaction Preferences. Survey results (left) and correlation matrix (middle, right) with respect to musical backgrounds, technology expertise and experience, and interaction styles or preferences, for (a) reViSiT and (b) the user's chosen sequencer.

In contrast to tracker users, music experience (e.g. performing, literacy) is generally more common amongst sequencer users, with the exception of piano skills, which are considerably more prevalent in tracker users (57% vs. 37%). It is notable that programs in which users tend to have performance backgrounds – especially *Ableton Live*, but also *FL Studio* and *REAPER* – correspond to those that show greater use of hardware interfaces, such as instruments and other controllers through microphones or MIDI. In the next section, these sequencers stand out as the more conducive to flow, and whilst such benefits can be attributed to properties of the software interface and notation, the directness and immediacy of hardware control can undoubtedly play a significant role. Indeed, the degree to which sequencers have been designed simply to support (and capture) the intimate musical interaction of a musician with their instrument might explain the limitations of some programs to support a focused, coherent, and fluid user experience in the absence of hardware. In this respect, the aforementioned programs also correspond to packages more explicitly targeting the desktop studio, based on close integration with limited hardware, in comparison to other sequencers that have traditionally catered for professional use, offering automation of extensive and complex electronic studios.

9.2 flow and experience

Analyses of flow were confounded by the influence of experience on the user's perception of a program. While it was expected to see more flow in *reViSiT* interaction, initial results did not bear this out, showing a significantly lower flow metric for *reViSiT* ($+0.42 \pm 0.065$) compared to other software ($+0.61 \pm 0.069$; $p < .05$).^{6,7} These figures, however, do not account for the inherent gap in users' program experience, between *reViSiT* and their chosen program – with most respondents indicating expertise with their chosen program, yet only recently being introduced to *reViSiT* (and tracking), as illustrated in Figure 1. The effect of this confounding variable is shown in Figure 2, where the flow metric is significantly higher in programs for users with relevant experience. Notably, the *reViSiT* tracker demonstrates a significantly higher flow metric, compared to sequencers, when only experienced users are considered ($p < .05$).⁸

⁶ Figures and tests quoted using mean average and 95% confidence level (intervals displayed).

⁷ Tested using a *one-tail, paired t-test* ($n=245$).

⁸ Tested using a *one-tail, unpaired t-test*, for users with experience 3 and 4 ($n_{reViSiT}=33$; $n_{sequencer}=177$).

Figure 1
User sample size by
program experience

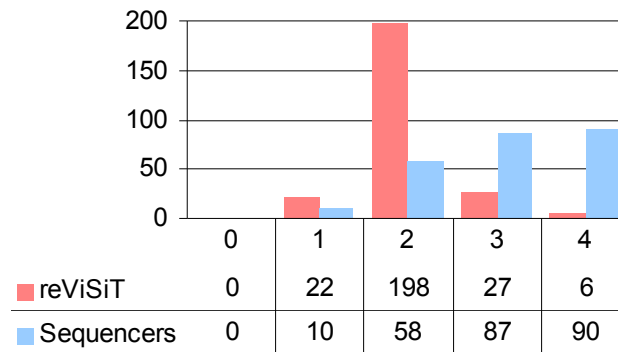
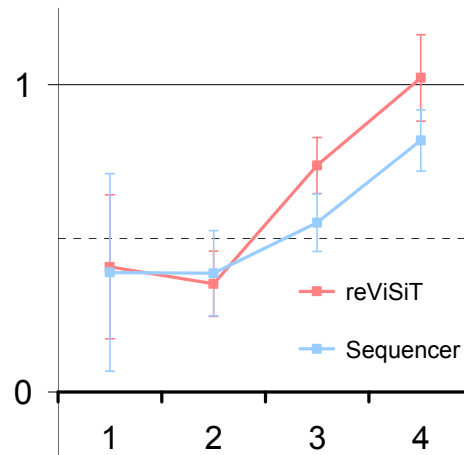


Figure 2
Flow metric by
program experience



To further illustrate the effect of experience within the sample, Figure 3 (a) shows the cognitive dimensions and flow profiles of the *reViSiT* data, together with a separate profile sampled exclusively from *reViSiT* experts (Experience 3 or 4). The overall flow metric for this sample is $+0.79 (\pm 0.194)$.

This figure and profile for *reViSiT* experts approaches that seen in other trackers (also given in Figure 3 (a)). Despite instructions to the contrary, some users elected to complete the sequencer section of the questionnaire in reference to another tracker – notably *Renoise*, self-styled as a “DAW with a vertical timeline sequencer”.⁹ However, this oversight enables the comparison of tracker and sequencer notations, and corresponding user experiences based on samples with comparable expertise, without relying on the restricted *reViSiT* sub-sample. At the same time, the wider *reViSiT* sample may provide insights into obstacles to flow at earlier stages of learning. Indeed, the extent to which the conditions for flow are improved by experience in these programs highlights the greater challenge for software designers, in crafting user experiences that enable flow for novice and first-time users.

⁹ See <http://www.renoise.com>.

Figure 3 (b) and (c) shows profiles generated from questionnaire responses, with regard to the cognitive dimensions of the notation and relative presence of flow components in the user experiences of popular music programs.¹⁰

The selection of programs in the survey is made by the users' personal preference, and thus corresponds to broadly positive impressions, based on extended experience, of their chosen program. As such, the profiles represent properties of the notation under skilled use, rather than those automatically available to new and novice users. This should be noted when considering the results in context with other uses of the cognitive dimensions framework, or other evaluation techniques for usability.

Direct comparisons with the results of *reViSiT* are not supported, due to the relative difference in the users' experience with each program (see 9.2). However, differences in relative trends within each survey are discussed as appropriate, and data from Renoise and other trackers provide an alternative basis for comparison of sequencing and tracking approaches, as illustrated in Figure 3 (b).

*common
creative profile*

A common cognitive dimensions profile emerges for the programs in the study. As the first application of this quantitative approach, no existing data exists to indicate whether this general profile is common amongst the music programs in the study, or whether it may be more universal. However, the characteristics identified broadly correspond to properties desirable in a notation designed for musical creativity (Blackwell and Collins, 2005; Duignan, 2007) and other *exploratory design* activities (Blackwell and Green, 2000; Blackwell et al., 2000), including:

- high *visibility* (ease of viewing and finding data)
- high *juxtaposability* (ease of comparing data)
- low *viscosity* (resistance to changing data)
- low *diffuseness* (conciseness, helping visibility and editing)
- high *role expressiveness* (ease of determining the role of objects)
- high *provisionality* (ease of sketching and experimentation)
- high *progressive evaluation* (ease of checking progress)
- high *consistency* (facilitating sense of control and learning)
- low *premature commitment* (freedom to change paths)

¹⁰ Dimensions are oriented so that higher values signify a generally positive impact on the user experience (e.g. low diffuseness), and allowing scores to be easily aggregated.

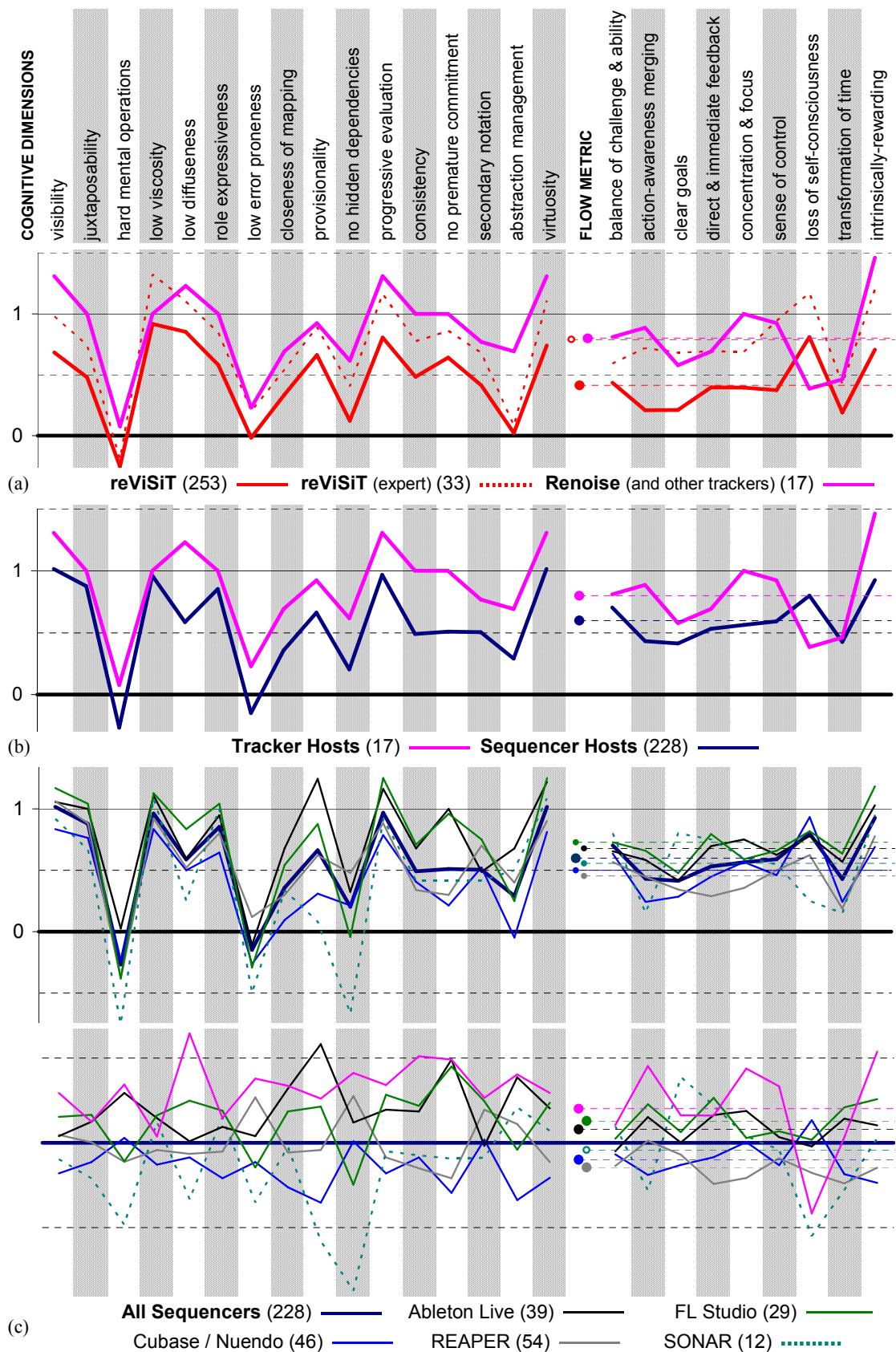


Figure 3 – Cognitive dimensions of notations (left) and flow component (right) profiles for music software, based on mean survey response (sample size), scored on a Likert scale (-2 strongly disagree; +2 strongly agree): (a) Tracker summary, with adjusted figure for reViSiT (compensating for user inexperience; see text); (b) Tracker and sequencer host comparison (excluding reViSiT); (c, top) Sequencer summary, average (**bold**) broken down by program; (c, bottom) Sequencer breakdown, including tracker hosts, as residuals from sequencer average.

trackers vs.
sequencers

At the same time, programs score lower with respect to generally undesirable properties, with neutral scores for *hard mental operations* (working things out in your head) and *error proneness* (ease of making mistakes). While creative individuals may be more tolerant (even welcoming) of mental challenges (see 3.3), the ease of making “careless mistakes” (Green and Petre, 1996) may signify a trade-off between allowing users creative freedom and exposing them to undefined program behaviour. However, results from these two dimensions also show a higher variance between users, compared to other dimensions, indicating that they are more susceptible to individual working styles.

Comparisons of tracker and sequencer responses, in Figure 3 (b), show a higher subjective opinion among tracker users, with regard to the majority of cognitive dimensions and flow components. *Diffuseness* stands out as a particular strength of the tracker, which benefits from its concise text-based notation. By contrast, *juxtaposability* is not significantly greater than that in sequencers, indicating one area where more flexible window layouts may benefit the user (e.g. by enabling side-by-side comparisons).

Role expressiveness similarly varies little between the two styles of music software; despite the heavy use of visual metaphors in sequencers, text labels and descriptions hold at least as much explanatory power in the tracker, for experienced users.¹¹ However, the large discrepancy in *consistency* can also be explained by a similar adherence to the use of text throughout the notation and interface, rather than the variety of graphics, texts, sub-devices, interaction styles, and input modalities in DAWs.

temporal scope
and premature
commitment

Sequencers also seem to involve more *premature commitment*, possibly due to the linear sequencing approach, where it is harder to insert or delete sections from the piece without considering the knock-on effect (e.g. to tempo, key, and other global settings). By contrast, the tracker divides the song into largely self-contained blocks (patterns) of music which are later placed in order. This explanation is supported by a similar trend in DAWs that also present a narrow temporal focus (e.g. patterns or loops), such as *FL Studio*, *Ableton Live* and trackers, which show less *premature commitment* than more traditional sequencers based on broader temporal scopes, using linear timelines and realtime capture.

flow in trackers

With regard to flow components, these properties of tracker notation contribute to improved *concentration & focus*, also facilitating *action-awareness merging*. Along with higher scores for *sense of control* and *intrinsic reward*, these results corroborate

¹¹ Though novice users may find more role-expressive notations easier to learn, initially.

similar findings in the interaction logs and video observations (see Chapter 6), arising from engaging and embodied interaction supported by learnt motor skills, high liveness, and focused use of a concise text-based notation, contained in a single, central view. The interaction style allows users to quickly sketch, play with, listen to musical ideas, and check their progress, which provides a source of motivation for continuing interaction, as users take satisfaction in their efforts and skill development.

*social creativity
in tracking*

The one exception to the trend is tracker's lower score for *loss of self-consciousness*, though *reViSiT*'s own score is more in line with sequencers. As discussed later, there are methodological issues with subjective measurement of this component, but the result may highlight a larger role of social interaction within the user community, where tracker musicians are more disposed to sharing their music and thus consider how it will be received by their peers (e.g. online, within the demoscene; see 2.2.2). As an extrinsic motivator, such influence has the potential to inhibit flow and creativity, which thrives on separation from the outside world (Amabile, 1983). At the same time, it is easy to avoid, retreat from, or only selectively engage in online participation.

flow in context

Figure 3 (c) shows deviations from the average sequencer profile for individual programs, highlighting the relative strengths and weaknesses of each, and the relative performance of trackers. *FL Studio* and *Ableton Live* notably stand out as the most conducive to flow, exhibiting more favourable cognitive dimensions profiles.

FL Studio

FL Studio's step-sequencer view (see Section 8.5), for example, enables greater conciseness (low *diffuseness*), by hiding details of musical events in the pattern display, but at the cost of increased *hidden dependencies*, which also reflects the prolific use of windows, leading to more hidden data (see section 8.4).

Ableton Live

Ableton Live benefits from its unified, single-window UI, supporting stronger *concentration & focus* compared to other windowed environments, including *FL Studio*. A relatively low *secondary notation* score suggests that the program would benefit from additional provisions that allow the user to sketch ideas outside the formal notation and procedures built-in to the program.

Cockos REAPER

REAPER, the most popular program in the survey, exhibited the lowest overall flow metric, especially with regard to feedback and focus. Judged with respect to the themes discussed in this chapter, its highly-configurable and scalable feature set (reflected in the high scores for *abstraction management* and *secondary notation*), coupled with a tiled, one-window interface, would be expected to provide better support for flow and creativity, among experienced

users. However, the extreme affordability of the program (free for non-commercial use) makes it a natural choice for first-time, amateur and hobbyist users. Thus, unlike the other programs surveyed, *REAPER* users averaged a lower level of expertise, which has been shown to impact the user's flow experience (see Section 6.4.2).¹² As with *reViSiT*, it thus becomes difficult to compare the data with that of other programs. Instead, the profile appears in the graph as a representation of the flow profile experienced by sequencer users at an earlier stage of development, which may highlight factors that UI designers can use to improve the learning experience – in this case, with respect to *progressive evaluation*, feedback and focus (see Section 6.3). Notably, in terms of process, *REAPER* is faithful to the linear timeline model seen in traditional sequencers, with its respective disadvantages.

In this survey of notation, *Cubase* (and *Nuendo*) consistently performs below the sequencer average, across most dimensions and flow components. Standout weaknesses inversely correlate with previously noted strengths of trackers – *provisionality*, *premature commitment*, and *progressive evaluation*. Earlier discussions relate such characteristics to software designed for later-stage creativity and goal-oriented productivity, rather than exploratory creativity (see Section 3.5). This partly reflects the software's origin and design focus: the electronic studio and professional audio production of recorded musical performances. However, the profile illustrates aspects of the program that function less well in the absence of studio hardware or performers, as is the case in the growing home and bedroom “desktop studio” market, or users looking for an integrated software-based authoring tool.¹³ In terms of flow, this is evident in the comparatively low score for *action-awareness merging*, where users have to stop more frequently to consciously think about what the appropriate actions are, and where more natural, reflexive, and skilled interaction is only enabled through external instruments or interfaces. In a similar vein, the low *abstraction management* score also indicates limitations in accommodating

¹² Indeed, reviews note that the extensive and advanced *REAPER* feature set can engender a significant learning curve, also reflected in the low *virtuosity* metric for the program. (Senior, 2009)

¹³ *Steinberg* markets a cut-down, loop-based sequencer, called *Sequel*, to the home market, which potentially addresses many of the interaction issues identified in this report – integrating all functionality into a single, tiled window, and providing dedicated features for working with shorter phrases of music. Unfortunately, the lack of support for the company's own plugin technology (VST) prevents *reViSiT* from running in the program, excluding it from the study. Indeed, the omission is indicative of what might be more widely perceived by users as a generally low ceiling to the software, impacting the long-term prospects for flow, and the opportunity to field test these innovations using expert and professional musicians, for inclusion in *Steinberg*'s professional line.

INTERACTION PREFERENCES	use program lots	like simple and easy	like powerful and advanced	enjoy program	most time on keyboard	avoid mouse	use keyboard a lot	use control surface	use microphone	use mouse wheel	use midi	both hands on keyboard	use keyboard shortcuts	use acoustic instrument	keyboard & mouse together	use midi controller	prefer keyboard	use mouse a lot	prefer mouse	most time on mouse	suffer discomfort	find program a chore
FLOW METRIC (reViSiT)	.24	.25	.29	.21	.23	.15	.23	n/a	n/a	-.04	-.09	.23	.17	n/a	.00	n/a	.13	-.05	-.24	-.09	-.21	-.19
FLOW METRIC	.36	.32	.26	.23	.17	.16	.15	.13	.12	.12	.11	.10	.10	.09	.08	.04	.03	.02	-.05	-.07	-.19	-.24
intrinsically-rewarding	.46	.32	.36	.40	.18	.13	.16	.06	.16	.15	.14	.06	.18	.05	.05	.03	.02	-.02	.03	-.03	.31	-.33
sense of control	.35	.32	.20	.27	.07	.14	.18	.07	.03	.15	.06	.01	.17	.06	.11	.04	.02	-.01	-.04	-.06	-.29	-.31
action-awareness merging	.21	.25	.14	.16	.10	.19	.18	.07	.05	.06	.09	.01	.07	.03	.09	.01	.09	-.03	-.09	-.07	.16	-.20
concentration & focus	.25	.25	.12	.08	.13	.08	.16	.13	.14	.05	.04	.11	.03	.09	.04	.01	-.03	-.07	-.04	-.09	-.12	-.14
direct & immediate feedback	.14	.20	.18	.04	.10	.05	.08	.09	-.02	.04	.08	.10	.08	.00	.04	.09	.04	.06	-.05	-.08	-.03	-.04
clear goals	.17	.12	.06	.04	.08	.14	.09	.06	.10	.12	.07	.08	.05	.19	.12	-.06	.04	-.05	-.10	-.02	-.05	-.06
balance of challenge & ability	.18	.15	.15	.11	.13	.11	.02	.07	.08	.01	.00	.08	.03	.09	.02	.02	-.03	.02	-.07	-.06	-.17	-.20
transformation of time	.16	.05	.14	.11	.14	.03	.04	.12	.15	.04	.03	.02	-.03	.00	-.05	.04	.02	.08	.00	.02	.06	-.01
loss of self-consciousness	.13	.19	.13	.11	.06	.02	-.04	.03	-.02	.06	.12	.09	-.01	.01	.03	.01	.01	.09	.03	.00	-.08	-.11
virtuosity	.31	.23	.26	.28	.08	.07	.03	.19	.21	.15	.09	-.01	.19	.13	.11	.12	-.01	.08	-.06	.04	-.17	-.16
visibility	.24	.30	.25	.25	.04	.15	.19	.09	.05	.17	.12	.09	.04	.08	.00	-.02	.03	.01	.04	-.12	.21	-.24
progressive evaluation	.26	.17	.27	.29	.12	.06	.11	.15	.16	.18	.14	.08	.13	.16	.06	.06	-.01	.03	.02	-.05	-.20	-.20
low viscosity	.17	.17	.16	.23	.07	-.03	.15	.11	.05	.17	-.01	.06	.12	.06	.06	.03	-.02	.08	-.01	-.15	-.23	-.28
role expressiveness	.19	.19	.24	.22	.00	.07	.07	.08	.17	.11	.10	-.01	.06	.10	.08	-.01	-.09	.06	.05	-.11	-.07	-.09
consistency	.14	.16	.12	.13	.09	.12	.13	.01	.09	-.02	-.01	.09	.08	.09	.01	.05	.00	-.06	.06	-.12	-.11	-.15
low diffuseness	.10	.12	.06	.09	.04	.03	.07	.07	.07	.08	.01	.03	-.06	.07	-.03	.04	-.01	-.01	.07	-.12	-.17	-.14
no premature commitment	.11	.11	.11	.11	.12	.11	-.03	.22	.11	.03	-.01	.08	.01	.04	-.05	.02	-.08	-.10	-.01	-.08	-.16	-.11
provisionality	.12	.14	.10	.13	-.09	.05	.05	.26	.12	.05	.07	-.07	.07	.07	.05	.07	-.06	.03	.03	-.19	-.25	-.10
closeness of mapping	.05	.14	.07	.03	-.08	.11	.12	.00	.07	-.07	-.01	.04	-.08	.08	-.09	-.01	-.06	-.06	.09	-.13	-.19	-.14
juxtaposability	.15	.19	.17	.18	-.03	.05	.12	.09	.08	.05	.05	-.02	-.06	.10	.04	.03	-.01	-.02	.05	-.08	-.13	-.16
secondary notation	.05	.01	.11	.08	.03	-.02	-.01	.06	.18	.05	.08	.01	.03	.22	.05	.02	.05	.03	.04	.07	-.18	-.21
abstraction management	.03	-.02	.07	.14	.11	.06	.07	.20	.15	.17	-.01	-.01	.10	.09	.04	.09	.05	-.05	-.12	.00	-.05	-.06
low error proneness	.15	.23	.16	.16	.01	.04	.05	.07	-.05	.05	.05	-.03	.21	-.02	.19	.14	-.04	.03	-.01	.02	-.06	-.18
no hidden dependencies	.17	.16	.19	.21	.05	.05	.16	.15	.10	.05	.04	-.01	.09	.18	.31	.09	-.02	.05	-.02	.05	.07	-.05
no hard mental operations	.19	.21	.10	.14	-.02	.19	.02	.12	.09	.03	.05	.04	.17	.05	.14	.07	.07	.03	-.09	.00	-.12	-.12

Table 2 – Correlation matrix between interaction preferences, flow and cognitive dimensions.

users looking to move beyond prescribed uses of the program. Like *reViSiT*, these *Steinberg* programs are yet to integrate a powerful end-user macro, scripting or programming capability, such as that seen in other programs.¹⁴

Together, the individual application profiles underscore the arguments for improved liveness, a stronger and narrower focus, and computer-based virtuosity, detailed earlier in this chapter, in contributing towards user experiences that support flow.

9.4 modelling flow with cognitive dimensions

Table 2 and 3 represent the correlations in responses from users of popular music software, together with the flow metric from the separate survey of their impressions of the *reViSiT* software. Again, only differences in relative trends between the results can be made, due to the relative difference in the users' experience with each program (see Section 9.2).

¹⁴ For example, *CAL Script* in *Cakewalk SONAR*, *Buzz machines* in *FL Studio* (and *Jeskola Buzz*), *Max/MSP* integration in *Ableton Live*, *JS* scripting in *REAPER*, *Lua* scripting in *Renoise*.

Table 2 illustrates how interaction preferences (e.g. for input devices or modes of working) relate to flow and cognitive dimensions. From these results, keyboard interaction is shown to be generally conducive to flow, especially when compared with equivalent results for the mouse, which appears to impede it. Interaction preferences related to keyboard use are consistently above those related to mouse use, when ranked by both the overall flow metric and with respect to individual components. Moreover, there exist negative correlations of flow and programs favouring the mouse (*prefer mouse*, $r=-.05$; *most time on mouse*, $-.07$). This is contributed to by low values for *action-awareness merging* ($r=-.09$) and *clear goals* ($r=-.10$), which may suggest the mouse's reliance on visual search encourages more conscious, reflective styles of thinking. Individually, these correlations are weak, but become increasingly relevant when considered in the context of other trends (*most time on keyboard*, $r=.17$), and echoed in related results (*avoid mouse*, $r=.16$). Indeed, the only properties of interaction ranked lower than those related to the mouse, are those inherently undesirable: *suffer discomfort* (e.g. RSI; $r=-.19$) and *find program a chore* ($r=-.24$).

While the generally high level of experience in this sample is likely a confounding factor that may bias results towards increased keyboard use (see Appendix G), it must be noted that, with very few exceptions, the interfaces of the programs studied here (i.e. sequencers) are primarily based on mouse-based interaction styles, such as WIMP and other direct manipulation-based approaches. At the same time, many of these programs are also designed to work with hardware, such as microphones, MIDI controllers, and control surfaces. While these input devices correlate positively with flow components, to the benefit of the wider user experience, they are largely designed to operate independently of, and away from, the main program and notation. This observation is reflected by positive correlations between *secondary notation* and both *use acoustic instrument* ($r=.22$) and *use microphone* ($r=.18$).

In the leftmost matrices of Figure 3, experience with a given program is seen to positively correlate with all flow components. The stronger correlations for *intrinsic reward* ($r=.31$) and *sense of control* ($r=.34$) fit with theories of creativity (see section 2.3), and the respective requirements of motivation and expertise. The related components of *concentration & focus* ($r=.29$) and *action-awareness merging* ($r=.25$) also show a relatively high correlation, the latter of which may indicate an increasing execution of tasks unconsciously, as actions are facilitated by learnt motor skills.

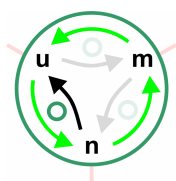
tracker-sequencer
migration

Interestingly, a user's impression of a program's *balance of challenge & ability* also seems to improve with experience ($r=.21$). This component was tested with respect to the level of challenge and the user's ability to meet that challenge. A positive correlation suggests that experience not only better equips users to meet the demands of the program, but also find new challenges. While this indicates the presence of high ceilings in modern music software, it might also indicate high entry thresholds, intimidating to novice users, such as performance skills in sequencers and literacy requirements in score editors and other trackers.

User responses were also correlated with respect to their stated experience with trackers, and notably that with the text- and keyboard-oriented *IT2* tracker (see 2.2.1). In this latter case especially but also in the general case, tracker experience correlates negatively with the user's perception of several flow components and cognitive dimensions in sequencer-style software. *IT2* users, forced to abandon the DOS-based tracker, seem to have had trouble finding a substitute music environment that suits their interaction style, or offers them sufficient support for flow ($r=-.05$). Negative correlations stand out for *progressive evaluation* ($r=-.19$), *premature commitment* ($r=-.18$) and *viscosity* ($r=-.12$), each of which can be related to the fast and free edit-audition cycle associated with most trackers. A similar relationship between flow and *closeness of mapping* ($r=-.12$) may also indicate a mismatch in these users' mental models of music, and that manifest in sequencers. Indeed, such challenging paradigm shifts are a consequence of the deeply-learned knowledge and skills associated with virtuosity, which itself may be harder to develop and maintain in fast-changing software environments.

the key role
of feedback

The rightmost matrices look at the correlations between individual cognitive dimensions and flow components, as well as the internal relationships between dimensions. The strongest correlation between a cognitive dimension and flow is *visibility* ($r=.53$), closely followed by *progressive evaluation* ($r=.51$). These respectively correspond to the availability of visual and musical feedback within the music editing environment, confirming their central role in the creative user experience, as well as the importance of liveness (see Section 4.2.4). In the survey of *reViSiT* use, these dimensions similarly ranked highest, but in reverse order: *progressive evaluation* ($r=.49$) and *visibility* ($r=.45$). This may reflect the greater emphasis given to musical feedback in the tracker interface, as well as the reduced use, detail, and richness of visual and graphical representations.



The final matrix shows the correlations between individual dimensions. Though a thorough analysis is largely beyond the scope of this thesis, this final matrix offers a way to empirically look at the relative orthogonality and granularity of dimensions (two of the acceptance criteria for new dimensions; see Blackwell *et al.*, 2001), and provides insight into the various trade-offs and dependencies between them. Moreover, the prominence of several recognised relationships between cognitive dimensions in this matrix also helps to validate the broader methodology and dataset, when looking at other aspects of interaction, such as flow.¹⁵

With regard to *virtuosity*, the matrix shows several moderate correlations, but none that suggest this new addition duplicates, or is otherwise fully-accounted for by, other dimensions. The dimension is tested against the phrase, “With time, I think I could become a virtuoso user of the system” – and, in line with flow’s *balance of challenge & ability* ($r=.42$), the quantity is not a measure of the difficulty of a notation, but the relative challenge faced by the user. Consequently, a simple notation designed for novices can score as well as a complex notation designed for experts. As will be shown, this dimension significantly contributes to the predictive power of cognitive dimensions when used as a model for designing flow-enabled user experiences.

To account for the internal interactions between different cognitive dimensions and identify the key dimensions that contribute to perceptions of flow in the user experience, the survey data was subjected to *multiple regression analysis*. Table 4 presents models produced by a stepwise regression analysis, using forward selection with *Mallows’ C_p* as a stopping rule to reduce the likelihood of overfitting the data.¹⁶ Three models are presented, respectively based on the surveys of *reViSiT* and other music software, and a combination of the two. Here, the relative difference in the experience present in each sample (see 9.2) is used to study the properties of notations that contribute to flow in

¹⁵ For example, *visibility* and *juxtaposition*, which are often combined as a single dimension, unsurprisingly show a relatively strong correlation (0.69). Indeed, there is a generally high correlation between dimensions based on visual properties of a notation. Dimensions targeting less tangible, more cognitive aspects, such as *error proneness* and *hard mental operations*, also seem to correspond. Moreover, it may also be significant that little interaction is shown between these two groups of dimensions. This could reflect the difficulty in relating visual and cognitive aspects of interaction, but may also reflect the relative polarity of dimensions. While dimensions are defined to be neutral descriptions of notational properties, the desirability of which is determined by context, some are more intrinsically good or bad, such as *error proneness*, *hard mental operations*, and *hidden dependencies*. Further analysis and explanation is left to future work.

¹⁶ The number of independent variables (16 dimensions) made an *all possible regressions* analysis too computationally intensive to be practical. However, the models presented were checked using an *all possible regressions* analysis on a subset of 12 variables, in which the 4 least significant terms were omitted (in all cases, $p > .5$), based on an initial *least-squares regression* analysis of all 16 dimensions.

novice and expert use, and also identify considerations relevant to bridging the divide, in designing multi-layered interfaces supporting both novices and experts. Individual factors are tested using a *student's t-test* (95% and 99% confidence levels are highlighted), and the models using *analysis of variance (ANOVA)*.

Regression	Multiple R	.699	ANOVA					.680 <th colspan="5">ANOVA</th> <td>.702<th colspan="5">ANOVA</th></td>	ANOVA					.702 <th colspan="5">ANOVA</th>	ANOVA								
	R ²	.489	df	7	213	220	Reg.	Res.	Total	df	6	195	201	SS	MS <td>3.839</td> <td>0.138</td> <th>df</th> <td>8</td> <td>414</td> <td>422</td> <th>SS</th> <td>MS</td>	3.839	0.138	df	8	414	422	SS	MS
	Adjusted R ²	.472	SS	26.98	28.17	55.15																	
	Standard Error	.364	MS	3.854	0.132																		
	Observations	221	F	29.15																			
	Mallows Cp	8.015	p	< .001																			

(a) reViSiT

(b) Other software

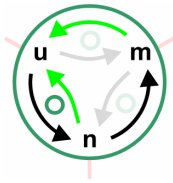
(c) Combined sample

Table 4 – Flow Models based on Cognitive Dimensions. Regression statistics, terms, and ANOVA results from modelling flow using forward selection stepwise regression, for each end-of-experiment survey, plus a model based on a combined sample. 95% (and 99%) significance levels are highlighted, for p-values in each model, and terms significant in all models.

All three models showed a strong goodness-of-fit, with R^2 and *adjusted R²* figures suggesting that between six and eight cognitive dimensions of the notation account for almost half the variation in the flow indicated by users. In models based on more dimensions, limited predictive power is gained, and the tendency for the model to overfit increases, with individual terms failing significance tests ($\alpha=.05$).¹⁷ The following paragraphs discuss the roles of individual dimensions in the model, with respect to flow (see Table 3) and findings from other analyses in this report. As appropriate, references are also made to the four design heuristics for supporting virtuosity (H_1 to H_4 , developed in Chapter 4) and their relationships to both specific cognitive dimensions (see Section 4.1.1) and corresponding findings in the survey. Table 5 illustrates these relationships, also highlighting dimensions appearing in the models above.

¹⁷ The maximum observed *adjusted R²* value was 0.486, based on a 12 variable model including the dimensions in Table 4 (c), plus *closeness of mapping* ($p=.099$), *hard mental operations* ($p=.125$), *hidden dependencies* ($p=.143$), and *juxtaposability* ($p=.810$).

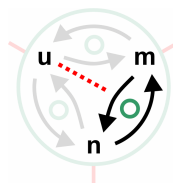
role of feedback



Three cognitive dimensions stand out as highly significant across all models: *visibility*, *progressive evaluation*, and *consistency*. *Visibility* (“easy to view and find parts of the music during editing”) reflects the importance of visual feedback and fast navigation around the notation, contributing to the user’s *sense of control* and enabling *action-awareness merging*, where the user is able to maintain focus on the task rather than peripheral distractions, mirroring the recommendations of H_4 , advocating “focused, modeless input”.

In music applications, *progressive evaluation* (“easy to stop and check my work while creating or editing it”) similarly highlights the importance of feedback, but from the musical domain, in the form of audio. This directly correlates with H_2 and the support of “rapid feedback cycles and responsiveness”. Through greater liveness (see Section 4.2.4), the causal effects of user actions are easily perceived, again contributing to a *sense of control*, but also allowing greater *concentration & focus* to rest as much with the actual music, as the abstract visual representation. Both dimensions are fundamental to the user’s understanding of what is going on in the program, and their music.

consistency
and learnability



The effective transparency of the notation enabled by rapid domain feedback improves the learnability of a program, where users can experiment with commands and features to understand their function. In this respect, the *consistency* (“Where aspects of the notation mean similar things, the similarity is clear in the way they appear”) allows users to transfer knowledge and expertise from one part of the program to another, and simplifies the overall management and learning of the system. By contrast, a lack of *consistency* can create unexpected program behaviour, leading to surprise and confusion that can harm the user’s *sense of control*, and potentially make them less confident and more self-conscious. This notably reiterates the need to provide “consistent output” (H_4), to facilitate “learning, memorisation, and prediction” (H_I).

beyond mastery

Two further dimensions are also significant in all three models, and likewise relate to learning and expertise: *virtuosity* and *abstraction management*. *Virtuosity* (“With time, I think I could become a virtuoso user of the system”) effectively gauges the *balance of challenge and ability*, which affects a user’s level of confidence to engage and experiment with the notation, and their motivation to develop expertise. As a creative individual becomes more expert, the level of formalism in a program’s notation can present a challenge; acting as a ceiling to their creativity, which is limited to paths more explicitly encoded in the representation, as

DESIGN HEURISTICS for VIRTUOSITY

H₁ Support learning, memorisation, and prediction (or “recall rather than recognition”)

H₂ Support rapid feedback cycles and responsiveness

H₃ Minimise musical (domain-) abstractions and metaphors

H₄ Support consistent output and focused, modeless input

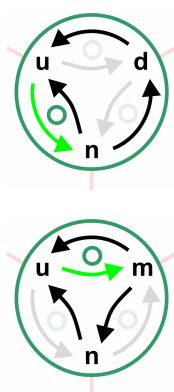
COGNITIVE DIMENSIONS			
●	●	●	consistency* Where aspects of the notation mean similar things, the similarity is clear in the way they appear.
●	●	●	low diffuseness The notation is concise and makes good use of space.
●	●	●	virtuosity* With time, it is possible for users to become expert with the notation.
●	●	●	closeness of mapping The notation matches how users would describe the music themselves.
●	●	●	hard mental operations When writing music with the notation, there are difficult things users have to work out in their head.
●	●	●	low viscosity* It is easy to go back and make changes to the music.
●	●	●	no premature commitment* Users can edit the song in any order, without having to think or plan ahead.
●	●	●	progressive evaluation* It is easy to stop and check work while creating or editing it.
●	●	●	provisionality Users can sketch things out and play around with ideas, without having to be too precise about the exact result.
●	●	●	secondary notation Users are able to make informal notes to themselves that aren't part of the music.
●	●	●	abstraction management* Users can do things to customise, adapt or use the program in ways its designer may not have intended.
●	●	●	role expressiveness* Within the overall scheme of my music, it is easy to see what each part is for.
○	●	●	visibility* It is easy to view and find parts of the music during editing.
●	●	●	juxtaposability It is easy to compare different parts of the music.
○	●	●	no hidden dependencies It is hard to see how different parts of the music relate to each other.
●	●	●	error proneness It is easy to make annoying mistakes.

Table 5 – the cognitive dimensions of notations, with brief descriptions of each dimension (as presented to users, see Appendix C for details; originally based on Blackwell and Green, 2003), also identifying relationships (positive ● and inverse ○) with the *design heuristics for virtuosity* (as detailed in Section 4.1.1). Dimensions marked * (also highlighted in green ●) further denote those likewise appearing in the regression model of Table 4.

envisaged by program designers. A notation’s *abstraction management* (“Users can do things to customise, adapt or use the program in ways its designer may not have intended”) determines the opportunity for users to appropriate the program as their own, for their own purposes, and extend its functionality and use beyond that envisaged by others. It gives those who know what they’re doing the ability to more precisely realise what they want (*clear goals*), and tackle higher challenges.¹⁸ To lower the

¹⁸ Overall motivation likely also benefits from taking a program beyond its apparent limits, which may also affect a user’s level of self-consciousness. However, rather than removing ego from the activity (as in *loss of self-consciousness*), motivation is gained with respect to an extrinsic source, the program designer (or other experts). This introduction of a synergistic extrinsic motivator (Amabile, 1996) complements the intrinsic motivation and reward upon which flow is based. That it occurs later in the user’s development, once they have built skill and confidence, may indicate that the individual has reached a stage in which

flow in tracking
and sequencing



combined model

threshold of creativity, H_3 advocates that notations “minimise musical and (domain-) abstractions and metaphors”, leaving users to develop their own understanding of the domain, moving from UIs based on “black boxes” to user-defined combinations of “simple primitives” (see p92-4). In this context, *abstraction management* is about providing mechanisms for the user to abstract complexity they have created themselves, which may include provisions for automation and scripting (see p217-8).

The role and importance of the remaining three dimensions in the models depend on context. The *reViSiT* model, fed by data on novice tracker users, includes terms for *viscosity* and *premature commitment*. The more general model, fed by data on experienced users of various music programs (notably sequencers), instead includes a term for *role expressiveness*. Respectively, these variables appear to reflect the strengths of the keyboard-controlled, pattern-based tracker and the visually-rich sequencer: trackers allow fast notation-based manipulation, sketching and experimentation with shorter passages of music (inset top-left, see Section 8.3), while sequencers are based on a linear timeline, song overviews and macroscopic editing (see Section 8.2), and more prominent representation of well-defined musical concepts and processes, including live performance and visual metaphors to traditional acoustic and electronic practices (inset left-bottom; see Section 2.1, Figure 2-2).

At the same time, each survey must be considered with respect to the confounding factor of experience – and how much the terms in each model indicate variables that respectively contribute to flow in novice and expert interaction. In the first instance, novices engaged in learning are likely to benefit from lower *viscosity* and lower *premature commitment*, enabling them to explore, play, and, experiment (“tinker”, Beckwith *et al*, 2006) with unfamiliar features of the notation, and easily backtrack or correct the mistakes beginners are liable to make. The absence of these factors in the model of more experienced sequencer use may support their larger role in earlier stages of learning.

The third model combines the questionnaire results of both surveys, in an attempt to produce a generalised flow model across a broader context of music software and its users, both novice and expert. While the emphasis (i.e. standardised coefficient, or *beta*) of each term is marginally reapportioned, the stepwise regression

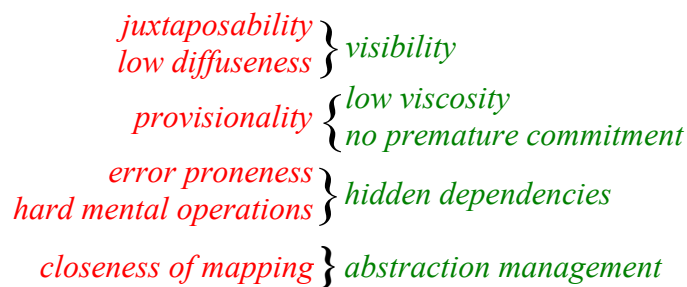
extrinsic motivation becomes significant, and where their creativity can be usefully modelled with respect to external and social factors (see Section 3.7).

process selects the same dimensions as before, including the five shared dimensions and all three context-specific dimensions. The higher variation of experience in the combined sample does not appear to significantly affect the terms selected by the stepwise regression process; no other dimensions are useful in capturing the influence of experience on flow.¹⁹

*non-orthogonality
between dimensions*

Of the eight absent dimensions, several are recognised to hold relationships with those already in the model, due to overlaps and trade-offs within the cognitive dimensions framework (Green and Petre, 1996). *Visibility*, for example, is often joined with *juxtaposability* as a single dimension, or cited as a trade-off against *hidden dependencies*. The interaction between dimensions reduces their supposed orthogonality, reducing their combined predictive power. In the stepwise selection process, such redundant variables are implicitly recognised and eliminated. Figure 4 highlights relevant interactions between dimensions where such redundancy occurs, most of which are also evident in Table 3.

Figure 4
Implicit relationships
in cognitive dimensions
between dimensions
eliminated and selected
in regression models
(Green & Petre, 1996;
see also Table 3)



*the individual
as a factor*

Additional variation in flow will also be influenced by factors beyond the notation, notably characteristics of the individual themselves, such as personality traits, aesthetic, environment, and also specific personal experiences within the program. The four remaining dimensions not used in the model also correspond to factors sensitive to individual contexts, habits, and experience. The related dimensions of *hard mental operations* and *error proneness* can arise from a breakdown in the *closeness of mapping*, where representations used in the program don't match the user's mental model. *Secondary notation*, distinct from the formal properties of the notation, also represents informal functionality that will be perceived and used very differently by different users.

¹⁹ An additional regression analysis was run with experience as a factor, to assess the overlap between this aspect of the individual and the properties of the notation. Using the combined sample, the term was significant ($p < .01$), but only a relatively low beta (.108) and marginal improvement in predictive power (*Adjusted R*² = .491) was observed.

Figure 5
Adjusted R^2 vs.
number of variables
in regression model.

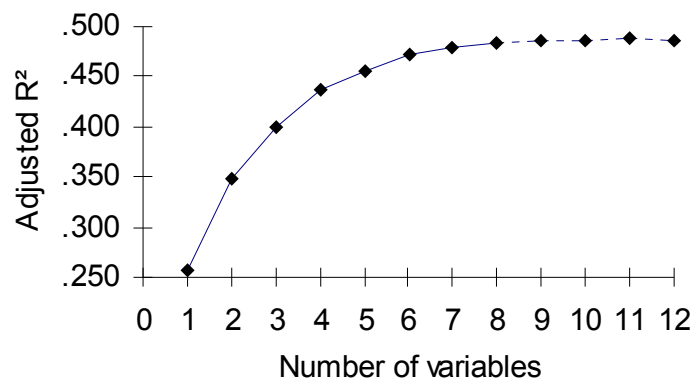


Figure 5 shows the diminishing growth of predictive power (*Adjusted R^2*) in models based on increasing numbers of cognitive dimensions, in the case of the combined sample. It illustrates the limited value of using more than 8 dimensions, but also highlights the significance of the five dimensions shared by all models. In the scenarios studied here, these five dimensions – *visibility*, *progressive evaluation*, *consistency*, *virtuosity*, and *abstraction management* – have proved less sensitive to context, and thus may represent the core properties desirable in notations supporting flow, in music software and possibly other creative contexts.



chapter ten **conclusion**

In the absence of an established canon of research into the composition process (Sloboda, 1999), this dissertation has drawn on psychological theories of creativity, in an effort to identify and address challenges in designing user experiences for computer-aided composition. As a study of real-world software and interaction, concepts of virtuosity and flow were explored in the context of tracking and sequencing user experiences, supported by a large-scale user study of over 1,000 tracker and sequencer users.

This final chapter reviews the approaches taken and findings made over the course of the research. Section 10.1 summarises the findings made through empirical investigations, the various methodologies of which are reviewed in Section 10.2. Section 10.3 then summarises theoretical offerings, developed to guide user studies and designed to provide a foundation for thinking about computer music experiences in general terms, to allow comparisons and cross-fertilisation across different music research contexts (e.g. performance, improvisation, composition) and applications (e.g. sequencers, trackers, score editors, live coding environments). Finally, Section 10.4 briefly discusses future directions for computer music research and development, suggested by the work.

Figure 1 (top)
Obsolet, by Voodoo (2004), a screenshot from a demo showcasing a history of the demoscene, and what one reviewer cites as an artist's impression of "deep hack mode".

10.1 summary of findings

Table 1 provides an overview of the specific findings of this research, with references to pages where they are discussed in the dissertation. Chapter 6 highlighted techniques that characterise expert tracker interaction, where well-learned motor skills and program knowledge are applied to enable focused interaction with the notation, frequently interwoven with playback, which maintains a high level of liveness, and a source of closure for recent work and impetus for further editing.

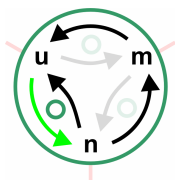
*tracking as a
bottom-up approach*

Trackers' bottom-up approach to editing music also benefits the learning experience by enabling composition using simple primitives, combinations of which can later be used to create more complex musical constructs. At the same time, high-level musical abstractions are only implicit in the notation and user's data, such that macroscopic editing (arrangement) and broader overviews of song structure are limited by the standard tracker UI, suggesting directions developers should look to innovate and may learn from DAW approaches (see, for example, Figure 2).

*the role of motor
skill in music*

The role and importance of motor skills was highlighted in Chapter 7, which concentrated on how users develop skill with the computer keyboard, to achieve fluency across interaction contexts within the tracker program. The importance of motor skill is well-established in music interaction (see Section 3.6), and as a critical factor of the expertise required for musical creativity (Section 3.5). In the sequencer, motor learning is supported in hardware interfaces such as MIDI controllers, but not available in software-based WIMP or direct manipulation interfaces that emphasise visual feedback, and often draw upon visual metaphors that afford physical interactions the mouse cannot support.

*rapid edit-
audition cycles*



In tracking, the keyboard cursor acts to anchor the user's focus and interaction, across both visual and aural contexts, with the notation and musical playback respectively. Commands such as *Play from Cursor (F7)*, in combination with fast cursor navigation, ensure that playback is readily available throughout editing, enabling fast edit-audition cycles (see inset). By emphasising domain (musical) feedback, composition progresses by listening and tinkering, rather than notational literacy or music performance skill, and thus enables experiential learning of music and composition (see Scripp *et al*, 1988; Folkestad, 1996).

Video Study (Chapter 6)

- use of set “postures” in different interaction contexts (6.1; Figure 2)
- “spot-on debugging”: quick listening episodes, poised to break into editing (6.2)
- “macro listening”: periods of more relaxed reviewing of work, for wider reflection and to combat tiredness caused by focused energetic interaction (6.2)
- “expand/explore”: bottom-up, exploratory creativity (6.2)
- experiential learning of composition technique (6.2)
- focused, energetic keyboard-based interaction over 8 hours, showing evidence of flow (6.3)

Keyboard and Motor Skill (Chapter 7)

- higher average interaction rate, but peak speed drops (due to tiredness and pacing) (7.1)
- indications of unconscious skill in musical entrainment of tempo (7.1)
- transition from mouse to keyboard interaction with experience (7.2; Figure 4)
- keyboard expertise a product of speed and knowledge, leading to fluency (7.3; Figure 8)
- navigation (cursoring) as basis for advanced skills (selections, playback) (7.3; Figure 9)
- simple actions can be learnt as atomic motor sequences (“finger macros”) (7.3; Figure 10)

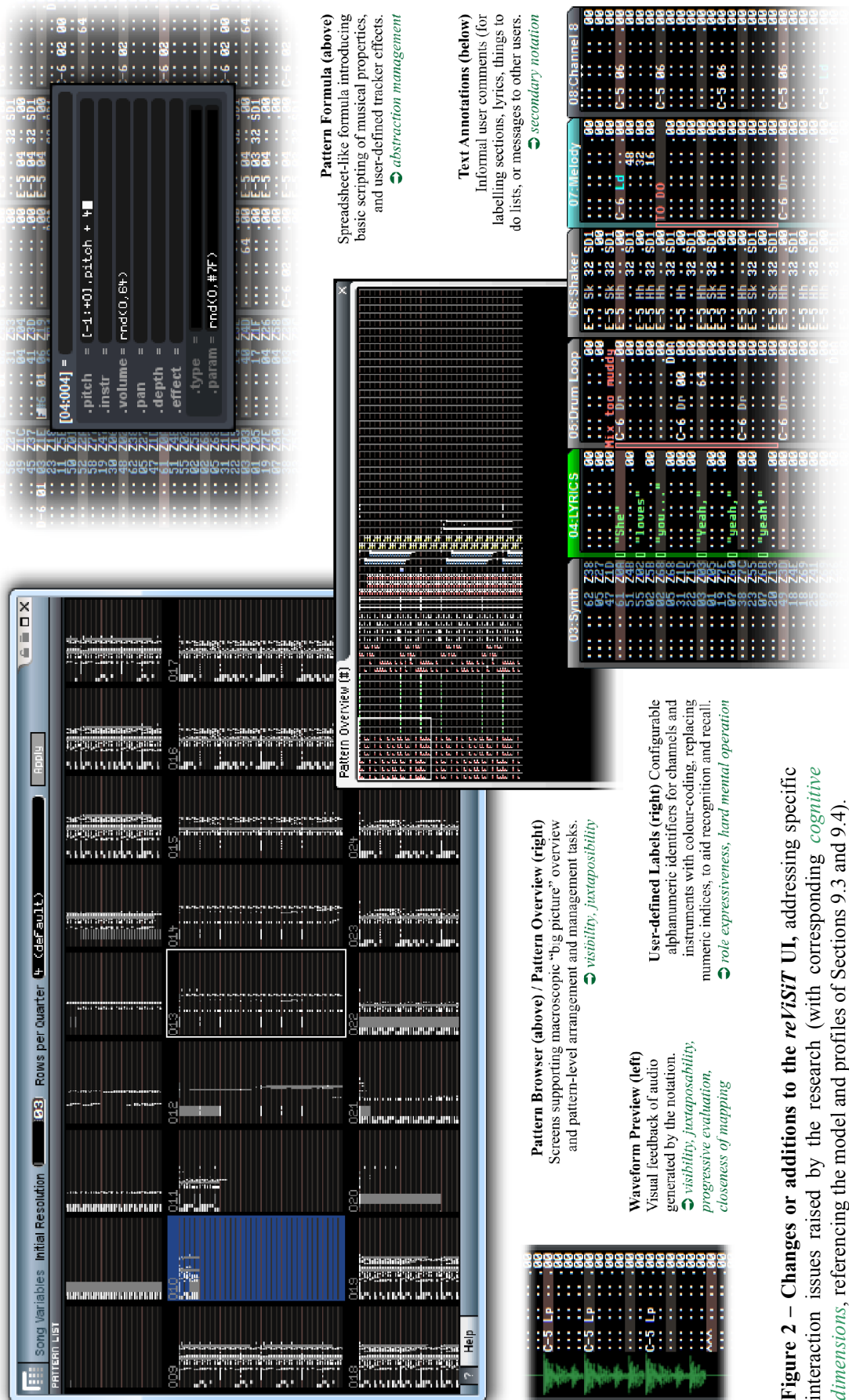
Focus and Feedback (Chapter 8)

- use of sequencer as tool for evaluation and refinement (8.1; Figure 1 to 3)
- 10-20 minute of preparation before active editing (and flow), worst for novices (8.1)
- over time, users tend towards having music playing for 2/3 of the time (Figure 4)
- trackers support rapid feedback, interwoven with editing activities (8.2)
- sequencer playback requires more preparation, suited to longer song playback (8.2)
- *liveness* appears more important to tracker experts than productivity (8.3; Figures 10 to 12)
- *liveness* (editing:feedback) as corollary of flow channel (boredom-anxiety) (8.3; Figure 12)
- windowed UIs can lead to cluttered workspaces, interfering with focus and requiring housekeeping (8.4; Figure 13 to 15)
- narrower editing scope (loops or patterns) benefits user focus (and flow) and *liveness* (8.5)

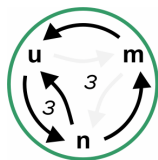
Flow and Cognitive Dimensions (Chapter 9)

- hardware control important in sequencer/DAW packages (9.1)
- flow in music software depends on user’s experience with program (9.2)
- music programs exhibit a similar cognitive dimension profile (9.3)
- flow can be usefully modelled using 5 to 8 cognitive dimensions (*Adjusted R*² = .483; 9.4)
 - core dimensions: *visibility*, *progressive evaluation*, *consistency*
(~ notation feedback, domain feedback, support for learning / *sense of control*)
 - other important dimensions: *virtuosity*, *abstraction management*
(~ *balance of challenge and ability*; “low threshold, high ceiling, wide walls”)
 - *viscosity*, *premature commitment*
(~ support for exploration, sketching, tinkering; “many paths”)
 - *role expressiveness*
(~ support for knowledge development / transfer)
- proposed *virtuosity* dimension useful as predictor of flow (~ *balance of challenge & ability*)
- tracker UIs exhibit more favourable flow profiles, compared to sequencers
(esp. *action-awareness merging*, *concentration & focus*, *sense of control*, *intrinsic reward*)
- sequencers also based on loops or patterns show improved support for flow (*Live*, *FL Studio*)

Table 1 – Summary of findings (with relative location within this report)



*maintaining
liveness*



maintaining focus

*flow and cognitive
dimensions*

Chapter 8 explored feedback mechanisms and the use and role of playback in both sequencers and trackers. In contrast to the close integration of playback in tracker interaction, which facilitated shorter, more frequent auditions of musical material, the more involved playback mechanisms of sequencers are best suited to auditions of longer passages of preset lengths (see Section 8.2). Section 8.3 shows that the liveness engendered by more frequent auditions during editing plays an important role in expert tracker interaction, even though greater productivity might be achieved by longer and more involved episodes of editing prior to playback. At the same time, the impact of interruptions created by playback likely decreases as users learn to employ shorter auditions, transition fluidly between editing and listening modes, and edit data during playback. The effect of these skills, as seen in expert tracker users, is to compromise between the stop-start, asynchronous edit-and-run (Level 2) liveness of visual notation editing and the direct, synchronous, realtime control of notation-less live performance (Level 4 liveness), enabling “direct involvement” and immersion in the musical domain (Leman, 2008). Arguably, the resultant interaction is comparable to Level 3 (edit-triggered) liveness, by virtue of the expert user’s own automaticity, in that they have learnt to reflexively trigger playback following a sequence of edits.

Chapter 8 also explored aspects of visual feedback that have the potential to distract or interrupt interaction, harming the user’s focus, sense of control, and flow of interaction. Inevitably, the more windows present in a workspace, the more housekeeping is required (Section 8.4). While this may appear to offer more flexibility for users to appropriate programs for their own use, there is little evidence that musicians are willing to invest the required attention. Moreover, this meta-management of the UI explicitly draws attention to the notational layer and away from the music, defeating attempts to make interaction as direct or transparent as possible.

Chapter 9 brings together many of the themes in earlier chapters, looking at the components of flow and properties of the notation, as present in tracker and sequencer use. Specific sequencer and tracker packages were profiled to identify favourable properties of digital music notations (Section 9.3), leading to a model of flow based on the *Cognitive Dimensions of Notations* framework (Green and Petre, 1996; see Section 9.4). The model suggests that flow mainly depends on a limited number of dimensions, corresponding to domain feedback (*progressive evaluation*), visual

feedback (*visibility, juxtaposability*), learning (*consistency, virtuosity, abstraction management*), and both fast and free editing (*low viscosity, low premature commitment*). These dimensions also echo calls for the provision of low thresholds (*virtuosity*), high ceilings (*abstraction management*) and wide walls (*low viscosity, low premature commitment*) in creative tools (Resnick *et al*, 2005), and contribute to conditions suiting exploratory creativity (see Section 3.1) and sketching (Sections 3.2 and 3.5).

Above all, this research has identified several areas and ways in which the current crop of music editing software can be improved to support more focused and rewarding user experiences, already seen in other areas of computer and music interaction. To this end, concepts of flow and virtuosity provide useful frameworks for articulating, modelling, and evaluating the motivational aspects and expertise supported by a tool, in facilitating the user's creativity. To progress beyond the productivity offered by current usability practices in authoring software, this research presents a case for UIs to support the user's development of virtuosity with the computer, as evident in the expert use of trackers. By identifying and generalising the properties of tracker interaction that facilitate learning, focus, sketching, and flow, it is hoped that the findings made here can be applied to other styles of music software that seek to provide comprehensive support for the user's creative process. The theories, models, and heuristics presented in Chapter 4 are reviewed in respect of this goal and the findings of the user studies presented here in Section 10.3.

10.2 methodological review

As a study of creative practice, this research has applied a variety of empirical methods to investigate the creative user experience, thus addressing limitations of individual approaches (Hewett *et al*, 2005). This section reviews the performance and utility of the methods used, from both research and engineering perspectives.

From a research perspective, this project has used a synthesis of qualitative and quantitative approaches, as well as idiographic and nomothetic approaches, in an effort to balance detail, validity, and reliability in investigations of musical creativity in composition, and overcome methodological challenges when such approaches are applied in isolation (see Chapter 3). The emergence of similar themes in each approach – including motor skill, focus, and feedback – underlines their importance in the user experience, but different analytic methodologies revealed different perspectives or granularities of detail in each.

The video study and discussions with a tracker composer (Chapter 6) provided a context for subsequent analyses of motor skill, focus, and feedback in the interaction logs of a large, distributed sample of tracker users from various backgrounds (Chapter 7-8). Significantly, the non-invasive logging of interaction enabled the study of real-world creativity, without interfering with the individual's creative process or intruding in their environment. In this capacity, the Internet has proven a powerful tool that can be instrumental in the remote observation of subjects in creativity research.

The large scale of the study allowed the habits and techniques observed in expert tracker practice to be examined in a more general population including less-experienced users; demonstrating how widespread such skills are, and also revealing how they develop with time. Longitudinal studies of tracker interaction were supplemented with surveys of a wider cross-section of music software use, including users of mainstream DAWs (sequencers, and loop- or sample-based triggers) and other trackers (Chapter 9). This report has also outlined how techniques used to study tracker interaction in detail might be applied to other activities and tools in digital music (Section 10.3; e.g. Section 4.2; Section 7.3), in developing taxonomies (e.g. Duignan, 2007) and models of computer music.

From an engineering perspective, evaluation techniques are also subject to practical considerations, which determine the benefit of using them in product development. Video studies, user surveys, and interaction logging are already employed by many companies to elicit user feedback. However, companies are rarely able to combine these approaches, target as large a sample of end-users, or afford the time to run longitudinal studies over months or years. Moreover, the *reViSiT* experiment (Chapter 5) benefited from the software's open, established, and enthusiastic community of tracker users (and wider demoscene culture), as well as the trust engendered by association with the University. Large corporations, by comparison, are likely to suffer more from the privacy concerns of users, which may restrict sample size or the detail of collectable data, but which might also be countered with larger incentives.

However, if a program is able to automatically collect data on its use, and that information can be automatically processed to highlight interaction issues, it could represent a low-cost method of providing ongoing feedback on the usability of a UI or program, and without requiring the active participation of the user.

Code libraries such as *iMPULS* (Chapter 5) could be established to collect and analyse common metrics, tailored to an activity or genre of software (e.g. music), minimising the initial cost of installation. Then, interactive visualisation applications such as *iMPULS|IVE* (Section 5.4) could be equipped with both preset and configurable filters, analyses, and visualisations that minimise the effort (and experience) required to probe data. As an engineering solution, visualisations of interaction data (see Appendix E) may also prove a more expedient tool, compared to statistic and quantitative methods, for quickly exploring user trends and debugging user experiences and interfaces, where the scientific rigour of statistical tests or development of quantitative metrics is less important. Other uses of interaction metrics, however, are explored in Section 10.4.

Indicators of flow experiences were also established in each methodology used. However, whereas quantitative analyses can be used to reveal corollaries of specific flow components – by, for example, looking at feedback use, interaction focus, or the continuity of action – investigation of the user’s mental state and subjective experience of flow (self-consciousness, awareness, sense of control, and perception of time) is ultimately only available from engagement with the user directly, through interviews, discussions, or user surveys. In practice, these more qualitative techniques may be sufficient to provide enough reliability and validity to inform design and engineering contexts (Sharp *et al*, 2007), if not the rigor to meet the requirements of scientific research (Weisberg, 1999; Csikszentmihalyi, 1999).

In this research, however, the benefit of combining several methodologies is that it enables us to refine the effectiveness and accuracy of simpler, low cost approaches, using insights from more extensive and involved analyses. Chapter 9, for example, demonstrated a simple application of the *Cognitive Dimensions of Notations* framework (Green and Petre, 1996) that drew upon the more detailed findings of previous chapters.

In addition to empirical findings, this dissertation discusses a theoretical foundation for considering notation-based music editing, and which enables empirical findings to be generalised and applied to other musical activities and applications. As summarised in Table 2, Section 4.1.1 offered a set of design heuristics for supporting virtuosity, based both on reviewed literature (Chapter 3) and themes examined in detail through user studies (Chapters 6-8, see table). In the process, both a descriptive model of music software interaction (Section 7.3) and a model of editing liveness (Section 8.3) were developed, which enabled comparisons across different tracker users, but which could also be extended to enable evaluations of other programs, including sequencers, score editors and other tools, in future research.

*flow, feedback
and liveness*

Section 4.2 presented a framework for illustrating liveness and flow properties of an interactive music system, defined as the product of one or more feedback cycles (Figure 3). Like Leman (2008), the model distinguishes between engagement with a notation and with the domain, but uses the iterative and cyclic properties of the creative process (Section 3.2) to describe systems as a synthesis of feedback from both the notation and the domain, demonstrating how liveness and flow can be supported in notation-based systems, such as trackers. By comparison, performance-based systems, such as sequencers, support liveness and immersion in music through realtime musical expression, which the computer automatically transcribes to notation.

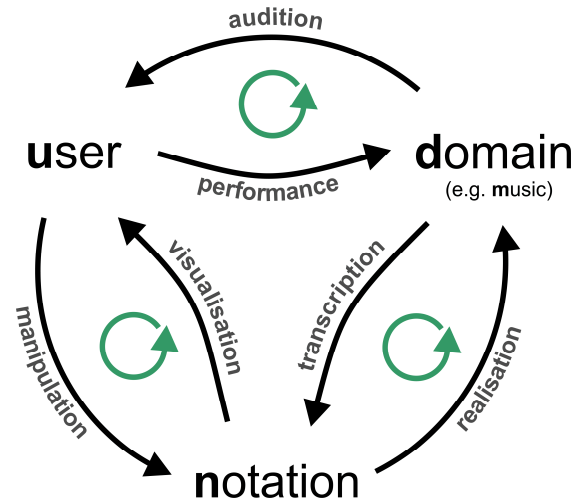
related themes (and studies) in tracker interaction

<i>H₁: Support learning, memorisation, and prediction</i> (or “recall rather than recognition”)	virtuosity, motor skill (Chapter 6 and 7)
<i>H₂: Support rapid feedback cycles and responsiveness</i>	liveness, music feedback (Chapter 6 and 8)
<i>H₃: Minimise musical (domain-) abstractions and metaphors</i>	process abstraction, UI primitives, metaphor ¹
<i>H₄: Support consistent output and focused, modeless input</i>	focus, visual feedback (Chapter 8, Section 4-5)

Table 2 – design heuristics for virtuosity (from Section 4.1.1) with references to related themes and studies in the report. See also: supporting findings in the study of flow in notation use (Chapter 9, p223).

¹ Though some aspects of this heuristic are explored (e.g. use of ‘simple primitives’ in notation-based systems vs. linear production processes, described in Chapter 2; bottom-up editing and tacit learning in trackers, Chapter 6; and through cognitive dimensions, *closeness of mapping*, *role expressiveness*, and *secondary notation*, Chapter 9), this research does not extensively explore the use of abstraction and metaphor in digital music. For detailed discussions of these issues, see the work of Duignan (2007; with Biddle, 2005; with Noble, Barr, and Biddle, 2004), which complements the themes studied here.

Figure 3
flow in notation use,
 adapted from the
 systems of musical
 flow model (presented
 in Section 4.2.3) for
 general application to
 creative practices



This effectively compartmentalises the creative process into a creation phase (the performance) using a hardware musical instrument, followed by a production phase (post-processing, mixing, arranging) using software, which has led to criticisms of the user experience in such software, as only supporting the later, incidental stages of creativity; the transcription or refinement of a musical idea (see Blackwell and Green, 2000; Blythe *et al*, 2007; Duignan, 2007).

*beyond realtime
 performance*

This research has demonstrated that a user experience can feel ‘live’ without relying on realtime interaction. As found in trackers and pattern- or loop-based sequencers, programs that couple fast editing of short passages with rapid feedback cycles are able to support a feeling of immersion and directness, while allowing time for users to think about and plan interactions. For novices, this relaxes the virtuosity required to engage in the musical domain, lowering the threshold for creativity (Scripp *et al*, 1988; Folkestad, 1996). At the same time, it gives experts the time to consider and experiment with more complex, advanced, and original musical solutions (compared to what is solvable in realtime performance or improvisation, see Johnson, 1980; Johnson-Laird, 1988; Sloboda, 1999; Burnard, 2007), raising the ceiling of creativity.

*towards flow in
 notation use*

In this way, users can manage the pace of interaction, allowing them to self-regulate the balance between challenge and ability, preserving a sense of control. Along with the creativity itself, the opportunity to discover, learn and develop virtuosity (including motor skills, composition technique, and musical knowledge) delivers an intrinsically-rewarding user experience. If a program can also maintain a user’s focus and concentration, without distractions or interruptions, flow experiences become possible, which this research has studied closely in the case of trackers, but also noted in some uses of sequencers.

10.4 future directions

From both theoretical and analytical perspectives, this work has attempted to lay a foundation for further studies of computer-aided composition. Within the field of digital music, research into tools for end-users is limited, and most work is driven by practice-based methods that cater for academic needs and aesthetics (Orio *et al*, 2001). This research, as well as that of Duignan (e.g. 2007), highlights significant areas for innovation and improvement in mainstream digital music practices and tools that would benefit from greater attention from established research communities (e.g. *NIME*, *ICMC*). Beyond the digital domain, this report also noted a paucity of studies, models, accounts, and theory concerning the creative processes, techniques, and tools of music composition, distinct from performance, improvisation, production, or other forms of musical creativity (see also Sloboda, 1999, 2005).

*musical content
analysis*

The quantity and depth of data collected during this research (see Section 5.2.2 and Appendix G) will support further analysis. The wide, online availability of compositions by tracker users coupled with program usage data from pitch entry and saved file summaries could, for example, provide information about the keys and harmonies used by composers at various stages of musical development – which may be used to examine how musical knowledge is self-taught, using interfaces that provide music feedback and allow tinkering with a notation.

*modelling other
musical activities*

The modelling approach described in Section 4.2.3 indicated how feedback, liveness, and flow are relevant in most digital music activities, and suggests that subsequent analyses of these phenomena (e.g. Chapters 7-9) can be replicated in other music programs and activities. Specifically, the questionnaire and model combining cognitive dimensions and flow, in Section 9.3 and 9.4, could be applied to other music editing environments, such as score editors, live coding, and visual programming environments.

*meta-interaction
and development
feedback*

As a learning environment, computer music tools such as trackers and those proposed by Scripp *et al* (1988) support a self-taught approach, and thus lack the instruction and direction traditionally imparted by tutors. Whilst online communities are a source of assistance (and extrinsic motivation), programs might be able to offer more dynamic, personally tailored solutions, using internal analysis of the user's interaction. The metrics used in Chapter 7, for example, describe how a user develops skill with respect to knowledge of the keyboard and program. In programs such as *reViSiT*, established thresholds and practices at different skill levels could be used by the program to not

*the computer
as music tutor*

only assess the user's stage of development, but also adapt program behaviour to emphasise or introduce new or unused features, shortcuts, or settings that suit the user's ability, and maintain an appropriate level of challenge.

If analyses are also extended to musical content, basic guidance concerning musical knowledge (tonality, harmony, rhythm) may also be deliverable.² The open-source nature of tracker songs (see Section 2.2.2) enables users to pick apart the music they listen to and admire, and future programs may be able to highlight the musical properties and devices used, and explain them to the user. Musical genres and styles, for example, are often distinguished by simple, common and easily-replicated tricks (Prochak and Prochak, 2001). However, from both a personal and cultural perspective, such mechanisms should be carefully considered for their capacity to influence the creativity of the individual.

*competing for
fun and practice*

Dynamic help or tutoring must also be careful not to interfere with the user's flow. While a program might intervene to maintain the *balance of ability and challenge*, the intervention itself must be carefully timed to minimise the disruption to the user's *sense of control* or *self-consciousness*. Lessons from managing flow in serious games (e.g. Sweetser and Wyeth, 2005; Michael and Chen, 2006) may be used, as might the idea of introducing game elements themselves. The role of motor skill has suggests learning should not be limited to following stepwise instructions of a tutorial, but should also be supported by exercises in fingering, command recognition, or timed set-task completion – possibly styled as a game with goals, targets, scores, and rewards. Activities could be designed to develop dexterity, knowledge, and familiarity with a program and its interactions, and motivated by competition with either oneself (through timing- and accuracy-based games encouraging self-improvement through intrinsically-motivated, deliberate practice) or one's peers (through advertising achievements with the community, based on sharing and flaunting exercise scores or interaction metrics).³

*from usability
to virtuosity...*

Lastly, this research has advocated a shift from usability to virtuosity, within the creative user experience, but Section 2.2.2 also notes how, in trackers, interaction skill plays a significant role beyond the UI. As researchers have argued (Csikszentmihalyi, 1999; see Section 3.1), it is also important to consider the social

² The program, for example, may detect that a user is implicitly favouring simpler musical styles – such as modal keys (e.g. C Major; the white keys of the piano, no accidentals), or 120bpm tempo, or a 4/4 time signature – and highlight features which may contribute to a broader musical palette.

³ The use of interaction metrics might also suggest how a program could connect users with suitable communities of practice; either artists of commensurable skill or experts to provide help or mentoring.

context of creative practice. The *demoscene* subculture represents a seldom-studied artistic community, but one that has also foreshadowed developments in wider digital creative practices and technology, such as online collaboration, digital communities, and music sharing (Botz, 2011). Trackers and the demoscene provide examples of how the role of virtuosity can contribute to a culture; acting as a source of extrinsic motivation for individuals, a pride in the creative process that complements the intrinsic motivation offered by the personal expression and satisfaction engendered by the creative product. The user's recognition of their skill is a factor worth considering in the design of a user experience, and a reason for further study of tracking and the demoscene subculture.

bibliography

- Albert, R. and Runco, M. "A History of Research on Creativity", in *Handbook of Creativity* (see Sternberg, 1999). pp. 16-34.
- Alty, J. 1995. "Navigating though Compositional Space: The Creativity Corridor", in *Leonardo*. 28(3):215-219. Cambridge, Massachusetts: MIT Press.
- Amabile, T. 1983. *The Social Psychology of Creativity*. New York: Springer-Verlag.
- Amabile, T. 1996. *Creativity in context: Update to The Social Psychology of Creativity*. Boulder Co.: Westview Press.
- Amabile, T. 2006. "How to Kill Creativity" in *Creative Management and Development* (ed. Henry, J.). London: SAGE Publications. Excerpt reprinted from *Harvard Business Review* (Sept., 1998). pp.18-24.
- Armstrong, N. 2006. *An Enactive Approach to Digital Musical Instrument Design*. PhD Thesis. Princeton University.
- Auh, M. 2000. "Assessing Creativity in Composing Music: Product-Process-Person-Environment Approaches", in *Proceedings of the 2000 National Conference of the Australian Association for Research in Education*. Available from: <http://www.aare.edu.au> [Last Accessed: Dec 2011]
- Auslander, P. 1999. *Liveness in a Mediatized Culture*. Abingdon, UK: Routledge.
- Bardzell, J. 2007. "Creativity in Amateur Multimedia: Popular Culture, Critical Theory, and HCI", in *Human Technology*, 3(1):12-33. Finland: Agora Centre, University of Jyväskylä.
- Barrett, M. 2005. "Children's communities of musical practice: some socio-cultural implications of a systems view of creativity in music education", in *Praxial music education: reflections and dialogues* (ed. Elliott, D. J.). pp.177-195. New York, Oxford University Press.
- Barrett, M. 2006. "Inventing songs, inventing worlds: the 'genesis of creative thought and activity in young children's lives'", in *International Journal of Early Years Education*. 14(3):201-220.
- Beckwith, L., Kissinger, C., Burnett, B., Wiedenbeck, S., Lawrance, J., Blackwell, A. and Cook, C. 2006. "Tinkering and gender in end-user programmers' debugging", in *Proceedings of CHI 2006*, pp. 231-240.
- Bederson, B. 2004. "Interfaces for Staying in the Flow", Editorial in *Ubiquity* (Sept 2004). ACM Press.
- Beilock, S., Carr, T., MacMahon, C., and Starkes, J. "When Paying Attention Becomes Counterproductive: Impact of Divided Versus Skill-Focused Attention on Novice and Experienced Performance of Sensorimotor Skills", in *Journal of Experimental Psychology: Applied*. 8(1):6-16. American Psychological Association.
- Bellotti, F., Berta, R., Gloria, A. and Primavera, L. 2009. "A task annotation model for SandBox Serious Games", in *Proceedings of CIG 2009*. Milano, Italy. IEEE Computing Society. pp. 233-240.
- Blackwell, A. 2002. "First Steps in Programming: A Rationale for Attention Investment Models", in *Proceedings of the IEEE Symposia on Human-Centric Computing Languages and Environments*, pp. 2-10. Washington, DC: IEEE Computer Society.
- Blackwell, A. 2006. "The Reification of Metaphor as a Design Tool", in *ACM Transactions on Computer Human Interaction (TOCHI)*, 13(4):490-530. Association of Computing Machinery.
- Blackwell, A., Britton, C., Cox, A., Green, T.R.G., Gurr, C., Kadoda, G., Kutar, M.S., Loomes, M., Nehaniv, C.L., Petre, M., Roast, C., Roes, C., Wong, A., and Young, R.M. 2001. "Cognitive Dimensions of Notations: Design Tools for Cognitive Technology", in Beynon, M., Nehaniv, C.L., and Dautenhahn, K. (Eds.) *Cognitive Technology*. LNAI 2117, pp. 325-341. Springer-Verlag.
- Blackwell, A.F., Church, L., Plimmer, B. and Gray, D. 2008. "Formality in sketches and visual representation: Some informal reflections", in *Sketch Tools for Diagramming* (ed. Plimmer, B. and Hammond, T.), workshop at *VL/HCC 2008*, pp. 11-18. IEEE Computer Society.
- Blackwell, A. and Collins, N. 2005. "The programming language as a musical instrument" in *Proceedings of PPIG 2005*, 120-130.
- Blackwell, A. and Green, T. 2000. "A Cognitive Dimensions questionnaire optimised for users", in *Proceedings of the PPIG 2000*, 137-152.

- Blackwell, A., Green, T. and Nunn, D. 2000. "Cognitive Dimensions of Notation Systems", presented at *ICMC 2000 Workshop on Notation and Music Information Retrieval in the Computer Age*. International Computer Music Association.
- Blackwell, A. and Green, T. 2003. "Notational Systems – the Cognitive Dimensions of Notations framework", in *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science*. pp.103-134. San Francisco: Morgan Kaufmann.
- Blythe, M., Light, A. and O'Neill, S. 2007. "Untitled: Emerging Cultural Forms in the Digital Age.", in *Human Technology*, 3(1):4-11. Finland: Agora Centre, University of Jyväskylä.
- Boden, M. 2004. *The Creative Mind: Myths and Mechanisms (2nd Edition)*. Abingdon, UK: Routledge.
- Botz, D. 2011. *Kunst, Code, und Maschine: Die Ästhetik der Computer Demoszene*. (in German) Bielefeld, Germany: transcript Verlag.
- Boyd, J. 1992. *Musicians in Tune: Seventy-five Contemporary Musicians Discuss the Creative Process*. New York: Simon and Schuster.
- Bryan-Kinns, N., Healey, P., and Leach, J. 2007. "Exploring Mutual Engagement in Creative Collaborations", in *Proceedings of C&C'07*. 223-232. Association of Computing Machinery
- Bryan-Kinns, N., and Hamilton, F. 2009. "Identifying Mutual Engagement", in *Behaviour & Information Technology*, 31(2): 101-125. London: Taylor & Francis.
- Burnard, P. and Younker, B. 2002. "Mapping Pathways: Fostering creativity in composition", in *Music Education Research*. 4:245-261.
- Burnard, P. 2007. "Routes to Understanding Musical Creativity", in *International Handbook of Research in Arts Education* (ed. Bresler, L.) pp.1199-1214. Dordrecht, NL: Springer.
- Burnett, R. 2003. "Multimedia: Back to the Future!" in *Perspectives on Multimedia: Communication, Media and Information Technology* (ed. Burnett, R., Brunstrom, A. and Nilsson, A.G.). Chichester, UK: John Wiley & Sons. pp. 1-16.
- Butler, J. 2008. "Creating Pedagogical Etudes for Interactive Instruments", in *Proceedings of New Interfaces for Musical Expression (NIME) 2008*. Genoa, Italy. pp. 77-80.
- Buxton, W. 1977. "A Composer's Introduction to Computer Music", in *Interface – Journal of New Music Research*, 16(2):57-72. Lisse: Swets and Zeitlinger.
- Buxton, W. and Myers, B. 1986. A study in two-handed input. In *Proceedings of CHI '86*. pp.321–326. NY: ACM Press.
- Byrne, C., MacDonald, R. and Carlton, L. 2003. "Assessing Creativity in Musical Compositions: Flow as an Assessment Tool", in *British Journal of Music Education*. 20(3):277–90.
- Campbell, D. 1960. "Blind variation and selective retention in creative thought as in other knowledge processes", in *Psychological Review*. 67:380-400. American Psychological Association Journals.
- Candy, L. and Edmonds, E. 2004. "Creative Expertise and Collaborative Technology Design", in *Proceedings of APCHI 2004*. Berlin: Springer-Verlag. pp. 60-69.
- Card, S., Moran, T., and Newell, A. 1983. *The Psychology of Human-Computer Interaction*. London: Lawrence Erlbaum Associates.
- Cascone, K. 2000. "The Aesthetics of Failure: "Post-Digital" Tendencies in Contemporary Computer Music", in *Computer Music Journal*. Cambridge, Massachusetts: MIT Press. pp. 12-18.
- Chafe, C. and O'Modhrain, S. 1996. "Musical Muscle Memory and the Haptic Display of Performance Nuance", in *ICMC Proceedings 1996*. International Computer Music Association. pp. 428-431.
- Chaffin, R. and Limieux, A. 2004. "General perspectives on achieving musical excellence", in *Musical Excellence* (ed. Williamon, A.). Oxford, UK: Oxford University Press. pp. 19-40.
- Chase, W. and Ericsson, K. 1981. "Skilled memory", in *Cognitive skills and their acquisition* (ed. Anderson, J.). Hillsdale, NJ: Lawrence Erlbaum Associates. pp. 141-190.
- Church, L., Nash, C., and Blackwell, A. 2010. "Liveness in Notation Use: From Music to Programming", in *Proceedings of PPIG 2010*. pp.2-11. Universidad Carlos III de Madrid.
- Clayton, M., Sager, R., and Will, U. 2004. "In time with the music: The concept of entrainment and its significance for ethnomusicology", in *ESEM Counterpoint*. 11:3-75.

- Cohen, A., Ivry, R. and Keele, S. 1990. "Attention and Structure in Sequence Learning", in *Journal of Experimental Psychology: Learning, Memory and Cognition*, 17:263-271.
- Colley, A., Banton, L., Down, J., and Pither, A. 1992. "An expert-novice comparison in musical composition", in *Psychology of Music*, 20(2):124-137. London: SAGE Publications.
- Collins, D. 2005. "A synthesis process model of creative thinking in music composition", in *Psychology of Music*. 33(2):193-216. London: SAGE Publications.
- Collins, D. 2007. "Real-time tracking of the creative music composition process", in *Digital Creativity*. 18(4):239-256. London: Routledge.
- Collins, K. 2008. *Game Sound*. Cambridge, Massachusetts, MIT Press.
- Collins, M. and Amabile, T. 1999. "Motivation and Creativity", in *Handbook of Creativity* (see Sternberg, 1999). pp.297-312.
- Cook, E., Teasley, S., and Ackerman, M.S. 2009. "Contribution, commercialization & audience: Understanding participation in an online creative community", in *Proceedings of ACM Group 2009*. pp.41-50.
- Cross, I. & Woodruff, G. E. 2009. "Music as a communicative medium", in *The Prehistory of Language, Volume 1* (eds. R. Botha and C. Knight). Oxford: Oxford University Press. pp. 113-144.
- Crutchfield, R. 1962. "Conformity and creative thinking", in *Contemporary Approaches to Creative Thinking* (ed. Gruber, H., Terrell, G., and Wertheimer, M.). NY: Atherton Press. pp. 120-140.
- Csikszentmihalyi, M. 1990. *Flow: The Psychology of Optimal Experience*. Australia: HarperCollins.
- Csikszentmihalyi, M. 1996. *Creativity: Flow and the Psychology of Discovery and Invention*. New York: Harper Perennial.
- Csikszentmihalyi, M. 1998. *Finding Flow: The Psychology of Engagement with Everyday Life*. Basic Books.
- Csikszentmihalyi, M. 1999. "Implications of a Systems Perspective for the Study of Creativity", in *Handbook of Creativity* (see Sternberg, 1999). pp. 313-338.
- Csikszentmihalyi, M. 2000. *Beyond Boredom and Anxiety: Experiencing Flow in Work and Play (25th Anniversary Edition)*. San Francisco: Jossey-Bass.
- van Dam, A. 1997. "Post-WIMP User-Interfaces", in *Communications of the ACM*, 40(2):63-67. Association of Computing Machinery.
- Davidson, L., and Welsh, P. 1988. "From collections to structure: the developmental path of tonal thinking", in *Generative processes in music: The psychology of performance, improvisation and composition* (ed. Sloboda, J.). pp.260-285. Oxford, UK: Clarendon Press.
- Dawkins, R. 1976. *The Selfish Gene*. NY: Oxford University Press.
- Deliège, I. and Harvey, J. 2006. "How can we understand creativity in a composer's work? A conversation between Irène Deliège and Jonathan Harvey", in *Musical creativity: multidisciplinary research in theory and practice* (ed. Deliège, I. and Wiggins, G.). Hove, UK: Psychology Press. pp. 397-404.
- DeMarco, T. and Lister, T. 1999. *Peopleware: Productive Projects and Teams (2nd Edition)*. NY: Dorset House.
- Desain, P., Honing, H., Rowe, R., Garton, B., Dannenberg, R., Jacobs, D., Pope, S.T., Puckette, M., Lippe, C., Settel, Z., and Lewis, G. 1993. "Putting Max in Perspective", in *Computer Music Journal*. 17(2):3-11. Cambridge, Massachusetts: MIT Press.
- Devoe, D. 1967. "Alternatives to Handprinting in the Manual Entry of Data", in *IEEE Transactions on Human Factors in Electronics*, 8(1):21-32.
- Dix, A., Finlay, J., Abowd, G., Beale, R. 1998. *Human-Computer Interaction*, Prentice Hall, Europe.
- Dix, A. 2005. "Upside-Down Vs and Algorithms-Computation Formalisms and Theory", in *HCI Models, Theories, and Frameworks: Toward A Multidisciplinary Science* (ed. Carroll, J.). San Francisco: Morgan Kaufman. pp. 381-430.
- Dourish, P. 2004. *Where the Action Is*. Cambridge, Massachusetts: MIT Press.

- Dowling, W. J. 1999. "The Development of Music Perception and Cognition", in *The Psychology of Music* (ed. Deutsch, D.). San Diego: Academic Press. pp. 603-626.
- Duignan, M. 2007. *Computer mediated music production: A study of abstraction and activity*. PhD thesis. NZ: Victoria University of Wellington.
- Duignan, M. and Biddle, R. 2005. "A Taxonomy of Sequencer User-Interfaces", in *Proceedings of ICMC 2005*. International Computer Music Association.
- Duignan, M., Noble, J., Barr, P. and Biddle, R. 2004. "Metaphors for Electronic Music Production in *Reason and Live*", in *Proceedings of APCHI 2004*.
- Edge, D. 2008. *Tangible Interfaces for Peripherals Interaction*. PhD Thesis. University of Cambridge. Available at: www.cl.cam.ac.uk/techreports/UCAM-CL-TR-733.pdf [Last Accessed: Nov 2011]
- Elliot, G.J., Jones, E., and Barker, P. 2002. "A grounded theory approach to modelling learnability of hypermedia authoring tools", in *Interacting with Computers*, 14:547-574. Elsevier Science.
- Emmerson, S. 2007. *Living Electronic Music*. Aldershot, UK: Ashgate.
- Ericsson, K., Krampe, R., and Tesch-Römer, C. 1993. "The Role of Deliberate Practice in the Acquisition of Expert Performance", in *Psychological Review*. 100(3):363-406. American Psychological Association Journals.
- Ericsson, K. and Kintsch, W. 1995. "Long-term working memory", in *Psychological Review*, 102(2): 211-245. American Psychological Association Journals.
- Ericsson, K. and Lehmann, A. 1996. "Expert and Exceptional Performance: Evidence of Maximal Adaptation to Task Constraints", in *Annual Review of Psychology*, 47:273-305. Annual Reviews.
- Feldman, D. 1999. "The Development of Creativity", in *Handbook of Creativity* (see Sternberg, 1999). pp. 169-188.
- Feinstein, J. 2011. "Optimal Learning Patterns for Creativity Generation in a Field", in *The American Economic Review*. 101(3):227-232. American Economic Association.
- Fencott, R. and Bryan-Kinns, N. 2010 "Hey Man, you're invading my Personal Space! Privacy and Awareness in Collaborative Music," in *Proceedings of NIME 2010*. pp. 183-203.
- Fischer, G. 2005. "Creativity and Distributed Intelligence", in *Creativity Support Tools* (see Shneiderman *et al*, 2005). pp. 71-73.
- Fischer, G., Giaccardi, E., Eden, H., Sugimoto, M., and Ye, Y. 2005. "Beyond Binary Choices: Integrating Individual and Social Creativity," in *International Journal of Human-Computer Studies (IJHCS)*. 63:4-5. Special Issue on Creativity (ed. Candy, L. and Edmond, E.). Duluth, MN: Academic Press.
- Fitts, P. and Posner, M. 1967. *Human Performance*. Oxford, UK: Brooks and Cole.
- Folkestad, G. 1996. *Computer Based Creative Music Making: Young People's Music in the Digital Age*. (Göteborg Studies in Educational Sciences 104). Gothenburg: Acta Universitatis Gothoburgensis.
- Fritz, B. and Avsec, A. 2007. "The experience of flow and subjective well-being of music students" in *Horizons of Psychology*. 16(2):5-17.
- Gabrielsson, A. 1999. "The Performance of Music", in *The Psychology of Music* (ed. Deutsch, D.). San Diego: Academic Press. pp. 501-602
- Gall, M., and Breeze, N. 2005. "Music Composition Lessons: The Multimodal Affordances of Technology", in *Educational Review*. 57(4): 415-433. London: Routledge.
- Gentner, D. and Nielsen, J. 1996. "The Anti-Mac Interface", in *Communications of the ACM*, 39(8):70-82. Association of Computing Machinery.
- Getzels, J. 1975. "Problem finding and the inventiveness of solutions", in *Journal of Creative Behavior*, 9:12-18. Creative Education Foundation.
- Getzels, J. and Csikszentmihalyi, M. 1976. *The Creative Vision: A Longitudinal Study of Problem Finding in Art*. NY: Wiley.
- Ginsborg, J. 2004. "Strategies for memorizing music", in *Musical Excellence* (ed. Williamon, A). Oxford, UK: Oxford University Press. pp. 123-140.
- Golann, S. 1962. "The creativity motive", in *Journal of Personality*. 30:588-600.

- Gordon, E. (1997). *Learning sequences in music: Skill, content and patterns; A music learning theory*. Chicago: GIA Publications.
- Graf, M. 1947. *From Beethoven to Beethoven: The Psychology of the Composing Process*. New York: Philosophical Library.
- Green, T. and Petre, M. 1996. "Usability Analysis of Visual Programming Environments: a 'cognitive dimensions' framework", in *Journal of Visual Languages and Computing*. 7: 131-174. Academic Press.
- Grout, D. and Palisca, C. 1996. *A History of Western Music (5th Edition)*. London: W.W.Norton & Co., Inc.
- Gruber, H. E. and Wallace, D. B. 1999. "The Case Study Method and Evolving Systems Approach for Understanding Unique Creative People at Work", in *Handbook of Creativity* (see Sternberg, 1999). pp. 93-115.
- Guérin, R. 2004. *Cubase SX/SL 2 Power!* Boston, MA: Course Technology.
- Guilford, J. 1950. "Creativity", in *American Psychologist*. 5:444-454.
- Guiard, Y. 1987. "Asymmetric division of labor in human skilled bimanual action: the kinematic chain as model", in *Journal of Motor Behavior*. 19(4):486-517.
- Hall, P. and Sallis, F. (eds.) *Twentieth-Century Musical Sketches*. Cambridge, UK: Cambridge University Press.
- Hallam, S. 2002. "Musical Motivation: towards a model synthesising the research", in *Music Education Research*. 4(2): 225-244. London: Routledge.
- Harding, J. 2010. "Image Line FL Studio 9", in *Sound On Sound*, March 2010. Cambridge, UK: SOS Publications.
- Harvey, J. 1999. *Music and Inspiration*. London: Faber and Faber.
- Healey, P. G. T. and Thiebaut, J.-B. 2007. "Sketching musical compositions", in *Proceedings of Cognitive Science (CogSci) Conference*, Nashville, USA.
- Hennessey, B. 1989. "The effect of extrinsic constraints on children's creativity while using a computer", in *Creativity Research Journal*, 2:151-168.
- Hewett, T., Czerwinski, M., Terry, M., Nunamaker, J., Candy, L., Kules, B., and Sytlan, E. 2005. "Creativity Support Tool Evaluation Methods and Metrics", in *Creativity Support Tools* (see Shneiderman *et al*, 2005). pp. 10-24.
- von Hippel, E. 2005. *Democratizing Innovation*. Cambridge, Massachusetts: MIT Press.
- Holtzblatt, K.A., Jones, S. and Good, M. 1988. "Articulating the experience of transparency: an example of field research techniques", in *SIGCHI Bulletin*, ACM Press, 20(2). pp. 46-48.
- Holtzblatt, K., Wendell, J., and Wood, S. 2005. *Rapid contextual design: A how-to guide to key techniques for user-centered design*. San Francisco: Morgan Kaufmann.
- Howe, M. 1999. "Prodigies and Creativity", in *Handbook of Creativity* (see Sternberg, 1999). pp. 431-448.
- Jackson, S., and Eklund, R. 2002. "Assessing flow in physical activity: The Flow State Scale-2 and Dispositional Flow Scale-2", in *Journal of Sport and Exercise Psychology*, 24:133-150.
- Jänke, L. 2006. "From cognition to action", in *Music, Motor Control and the Brain* (ed. Altenmüller, E., Wiesendanger, M. and Kesselring, J.). Oxford, UK: Oxford University Press. pp. 25-38.
- Jennett, C., Cox, A., Cairns, P., Dhoparee, S., Epps, A., Tijs, T. and Walton, A. 2008. "Measuring and Defining the Experience of Immersion in Games", in *International Journal of Human-Computer Studies*, 66(9):641-661.
- John, B. and Gray, W. 1995. "CPM-GOMS: an analysis method for tasks with parallel activities", in *CHI'95 Conference Companion*. 393-394.
- Johnson, D. 1980. "Beethoven's early sketches in the 'Fischhof Miscellany'", in *Berlin Autograph*. 28(1-2). Michigan: UMI Research Press. pp. 511-516.

- Johnson, W.L., Wilhjalmsson, H., and Marsella, S. 2005. "Serious Games for Language Learning: How Much Game, How Much AI?", in *Proceedings of the Artificial Intelligence in Education*. pp.306-313. Amsterdam: IOS Press.
- Johnson-Laird, P. 1988. "Freedom and constraint in creativity", in *The nature of creativity: Contemporary psychological perspectives* (ed. Sternberg, R.). pp.202-219.
- Jones, M. 1998. "Creating Electronic Learning Environments: Games, Flow, and the User Interface", in *Proceedings of the AECT 1998*. St. Louis, MO. pp. 205-214.
- Jordà, S. 2001. "New Musical Interfaces and New Music-making Paradigms" in *Proceedings of New Interfaces for Musical Expression, CHI 2001*. Association for Computing Machinery. pp. 1-5.
- Jordan, P. 2002. *Designing Pleasurable Products*. CRC Press.
- Junglas, I. and Steel, D. 2007. "The Virtual Sandbox", in *DATA BASE for Advances in Information Systems*. 38(4):26-28.
- Kirsh, D. and Maglio, P. 1994. "On distinguish epistemic from pragmatic action", in *Cognitive Science*. 18:513-549.
- Kitzmann, A. 2003. "Transparency, Standardization and Servitude: the Paradoxes of Friendly Software", in *Perspectives on Multimedia* (ed. Burnett, R., Brunstrom, A., and Nilsson, A.). Chichester, UK: John Wiley & Sons. pp. 41-54.
- Knörig, A. 2006. *Free the body and the mind will follow: : An investigation into the role of the human body in creativity, and its application to HCI*. Diplom-Medieninformatiker (FH) Thesis. University of Applied Sciences, Wedel. Available at: <http://www.andreknorig.de> [Last accessed: Dec 2011]
- Koestler, A. 1964. *The Act of Creation*. London: Hutchinson & Co.
- Kris, E. 1952. *Psychoanalytic Exploration in Art*. NY: International Universities Press.
- Kratus, J. 1989. "A Time Analysis of the Compositional Processes Used by Children Ages 7 to 11" in *Journal of Research in Music Education*. 37(1):5-20. MENC: The National Association for Music Education.
- Kubie, L. 1958. *The Neurotic Distortion of the Creative Process*. Lawrence: University of Kansas Press.
- Leman, M. 2008. *Embodied Music Cognition and Mediation Technology*. Cambridge, Massachusetts: MIT Press.
- Lerdahl, F. 1988. "Cognitive Constraints on Composition Systems", in *Generative Processes in Music: The Psychology of Performance, Improvisation, and Composition* (ed. Sloboda, J.). Oxford, UK: Oxford University Press. pp. 231-259.
- Lerdahl, F. and Jackendoff, J. 1983. *A Generative Theory of Tonal Music*. Cambridge, Massachusetts: MIT Press.
- Linson, A. 2011. "Unnecessary Constraints: A Challenge to some Assumptions of Digital Musical Instrument Design", in *Proceedings of ICMC 2011*. International Computer Music Association. pp. 421-424.
- Lubart, T. 2005. "How can computers be partners in the creative process: Classification and commentary on the Special Issue", in *International Journal of Human-Computer Studies*. 63:365-369.
- Lumsden, C. 1999. "Evolving Creative Minds: Stories and Mechanisms", in *Handbook of Creativity* (see Sternberg, 1999). pp. 153-168.
- MacDonald, Ronan. 2007. "Trackers!", in *Computer Music*, 113:27-35. Bath, UK: Future Publishing, Ltd.
- MacDonald, Raymond. Byrne, C. and Carlton, L. 2006. "Creativity and flow in musical composition: an empirical investigation", in *Psychology of Music*, 34(3):292-306. Society for Education, Music, and Psychology Research.
- MacKenzie, S. 2003. "Motor Behavior Models for Human-Computer Interaction", in *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science*. pp.27-54. San Francisco: Morgan Kaufmann.
- Magnusson, T. and Mendieta, E. 2007. "The Acoustic, the Digital and the Body: A Survey on Musical Instruments", in *Proceedings of NIME 2007*. pp. 94-97.





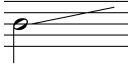
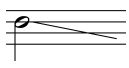
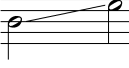

- Mandler, G. 1995. "Origins and consequences of novelty", in *The Creative Cognition Approach* (ed. Smith, Ward, and Finke). pp.9-25. Cambridge, Massachusetts: MIT Press.
- Manovich, L. 2001. *The Language of New Media*. Cambridge, Massachusetts: MIT Press.
- Martindale, C. 1999. "Biological Bases of Creativity", in *Handbook of Creativity* (see Sternberg, 1999). pp. 137-152.
- Maslow, A. 1963. "The Creative Attitude" in *Structuralist* (ed. Bornstein). 3:4-10.
- Maslow, A. 1968. *Towards a psychology of being (2nd Edition)*. NY: D. Van Nostrand Co.
- May, R. 1975. *The Courage to Create*. NY: W.W. Norton & Co.
- Mayer, R.E. 1999. "Fifty Years of Creativity Research", in *Handbook of Creativity* (see Sternberg, 1999). pp. 449-460.
- McBride, N. and Brown, S. *Toward Computer Systems to Support Creativity*. Available at: [http://www.cse.dmu.ac.uk/~nkm/PAPERS/Toward Computer Systems to Support Creativity.pdf](http://www.cse.dmu.ac.uk/~nkm/PAPERS/Toward%20Computer%20Systems%20to%20Support%20Creativity.pdf) [Last Accessed: Dec 2011]
- McCullough, M. 1996. *Abstracting Craft: The Practiced Digital Hand*. Cambridge, Massachusetts: MIT Press.
- McLean, A. and Wiggins, G. 2011. "Texture: Visual Notation for Live Coding of Pattern", in *Proceedings of ICMC 2011*. International Computer Music Association. pp. 621-628.
- Michael, D. and Chen, S. 2006. *Serious Games: Games that Educate, Train, and Inform*. Course Technology.
- Miller, R. 1968. "Response time in man-computer conversational transactions", in *Proceedings of AFIPS Spring Joint Computer Conference*. 33:267-277.
- Millward, S. 2005. *Fast guide to Cubase SX (3rd Edition)*. UK: PC Publishing.
- Mohamed, F., and Fels, S. 2002. "KEYed user interface: Tools for expressive music production", in *ICMC Proceedings 2002*. pp.88-91. International Computer Music Association.
- Moran, J., and Liou, E. 1982. "Effects of reward on creativity in college students of two levels of ability" in *Perceptual and Motor Skills*, 54:43-48.
- Nabavian, S., and Bryan-Kinns, N. (2006). "Analysing Group Creativity: A Distributed Cognitive Study of Joint Music Composition", in *Proceedings of Cognitive Science*, pp. 1856-61.
- Nash, C. 2004. *VSTrack: Tracking Software for VST Hosts*. MPhil Thesis. Available from: <http://vstrack.nashnet.co.uk>. [Last Accessed: Dec 2011]
- Nash, C. 2008. "Realtime Gestural Control and Representation of Musical Polytempi", in *Proceedings of New Interfaces for Musical Expression (NIME) 2008*. Genoa, Italy. pp. 28-33.
- Nash, C. and Blackwell, A. 2011. "Tracking Virtuosity and Flow in Computer Music", in *Proceedings of ICMC 2011*. International Computer Music Association. pp. 575-582.
- Nash, C. and Blackwell, A. 2012. "Liveness and Flow in Notation Use", in *Proceedings of New Interfaces for Musical Expression (NIME) 2012*. Ann Arbor, MA. pp. 28-33.
- Nash, C. and Blackwell, A. 2012. "Flow of creative interaction with digital notations", in *Oxford Handbook of Interactive Audio* (in press). Oxford, UK: Oxford University Press.
- Newell, A. 1990. *Unified Theories of Cognition*. Harvard University Press.
- Nickerson, R. 1999. "Enhancing Creativity", in *Handbook of Creativity* (see Sternberg, 1999). pp. 392-430.
- Nielsen J. and Molich R. 1990. "Heuristic evaluation of user interfaces", in *Proceedings of the ACM CHI '90*. Association of Computing Machinery. pp. 249-256.
- Nielsen, J. 1993. *Usability Engineering*. Cambridge, Massachusetts: AP Professional.
- Nielsen, J. 1994. *Response Times: The Three Important Limits*, available from: <http://www.useit.com/papers/responsetime.html> [Last Accessed: May 2010].
- Norman, D. 1988. *The Psychology of Everyday Things*. NY: Basic Books.
- Norman, D. 1993. *Things That Make Us Smart*. NY: Basic Books.





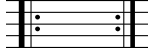


- Norman, D., and Draper, S. (eds.) 1986. *User Centered System Design: New perspectives on human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Orio, N., Schnell, N. and Wanderley, M. 2001. "Input Devices for Musical Experience: Borrowing Tools from HCI", presented at the *ACM CHI'01 New Interfaces for Musical Expression Workshop*. Association of Computing Machinery.
- Paradiso, J. and O'Modhrain, S. 2003. "Current Trends in Electronic Music Interfaces", in *Journal of New Music Research*, 32(4). London: Routledge. pp. 345-349.
- Perkins, D. 1981. *The mind's best work*. Cambridge, Massachusetts: Harvard University Press.
- Plucker, J.A. and Renzulli, J.S. 1999. "Psychometric Approaches to the Study of Human Creativity", in *Handbook of Creativity* (see Sternberg, 1999). pp. 35-61.
- Polgár, T. 2008. *FREAX: The Brief History of the Computer Demoscene*. Winnenden, Germany: CSW-Verlag.
- Policastro, E. and Gardner, H. 1999. "From Case Studies to Robust Generalizations: An Approach to the Study of Creativity", in *Handbook of Creativity* (see Sternberg, 1999). pp. 213-225.
- Polson, P.G., Lewis, C., Rieman, J., and Wharton, C. 1992. "Cognitive walkthroughs: A method for theory-based evaluation of user interfaces" in *International Journal of Man-Machine Studies*. 36: 741-773.
- Prochak, M. and Prochak, T. 2001. *How to get the sound you want*. Sanctuary Publishing.
- Reitman, W. 1965. *Cognition and Thought*. NY: Wiley.
- Resnick, M., Myers, B., Nakakoji, K., Shneiderman, B., Pausch, R., Selker, T and Eisenberg, M. 2005, "Design Principles for Tools to Support Creative Thinking", in *Creativity Support Tools* (see Shneiderman *et al*, 2005). pp. 25-36.
- Rittel, H. and Webber, M. 1984. "Dilemmas in a General Theory of Planning", in *Developments in Design Methodology* (ed. Cross, N.). Chichester, UK: John Wiley & Sons. pp. 135-144.
- Robinson, A. (ed.). 2007. "reViSiT", in *Computer Music Special: Freeware 2007*, 22:90-91. Bath, UK: Future Publishing.
- Rogers, Y. 2006. "Moving on from Weiser's Vision of Calm Computing: Engaging UbiComp Experiences", in *Proceedings of UbiComp 2006*. Berlin: Springer-Verlag. pp. 404-421.
- Rothermel, K., Cook, C., Burnett, M., Schofield, J., Green, T., and Rothermel, G. 2000. "WYSIWYT Testing in the Spreadsheet Paradigm: An Empirical Evaluation", in *Proceedings of ICSE'00*. pp.230-9.
- Rubin, L. 1968. "Creativity and the Curriculum", in *Teaching for Creative Endeavour: Bold New Venture* (ed. Michael, W.). pp.74-89. Bloomington: Indiana University Press.
- Runco, M. and Sakamoto, S. 1999. "Experimental Studies of Creativity", in *Handbook of Creativity* (see Sternberg, 1999). pp. 62-92.
- Rutkowski, C. 1982. "An introduction to the Human Applications Standard Computer Interface, Part I: Theory and principles", in *Byte*, 7 (10):291-310.
- Joel Ryan. 1991. "Some remarks on musical instrument design at STEIM", in *Contemporary Music Review*, 6(1):3-17.
- Sawyer, R.K. 1995. "Creativity as mediated action: A comparison of improvisational performance and product creativity", in *Mind, Culture, and Activity*, 2:172-191.
- Sawyer, R.K. 2006. "Group creativity: musical performance and collaboration", in *Psychology of Music*, 34(2): 148-65. London: SAGE Publications.
- Sellen, A. and Harper, R. 2003. *The Myth of the Paperless Office*. Cambridge, MA: MIT Press.
- Schubert, G. and Sallis, F. "Sketches and sketching", in *Twentieth-Century Musical Sketches* (ed. Hall, P. and Sallis, F.). pp.5-16. Cambridge, UK: Cambridge University Press.
- Scripp, L., Meyaard, J., and Davidson, L. 1988. "Discerning Musical Development: Using Computers to Discover What We Know" in *Journal of Aesthetic Education*. 22(1):75-88. University of Illinois Press.
- Shamrock, M. 1997. "Orff-Schulwerk: An Integrated Foundation", in *Music Educators Journal*. 83(6):41-44. MENC: The National Association for Music Education.




- Sharp, H., Rogers, Y., Preece, J. *Interaction Design: Beyond Human-Computer Interaction (2nd Edition)*. Chichester, UK: John Wiley & Sons.
- Shehan, P. 1986. "Major Approaches to Music Education: An Account of Method", in *Music Educators Journal*. 72(6):26-31. MENC: The National Association for Music Education.
- Shneiderman, B. 1983. "Direct Manipulation: A Step Beyond Programming Languages", in *Computer*, August 1983:57-69, Washington, DC: IEEE Computer Society.
- Shneiderman, B. 1996 "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations", in *Proceedings of the IEEE Symposium on Visual Languages*, pp. 336-343.
- Shneiderman, B. 2002. *Leonardo's Laptop: Human Needs and the New Computing Technologies*. Cambridge, Massachusetts: MIT Press.
- Shneiderman, B., Fischer, G., Czerwinski, M., Myers, B., and Resnick, M. 2005. *Creativity Support Tools*, NSF Workshop Report. Available from: <http://www.cs.umd.edu/hcil/CST/> [Accessed: Apr. 2010].
- Shneiderman, B. and Plaisant, C. 2005. *Designing the User Interface (4th Edition)*. Addison Wesley.
- Simonton, D. 1994. "Computer Content Analysis of Melodic Structure: Classical Composers and Their Compositions", in *Psychology of Music*. 22: 31-43. London: SAGE Publications. pp. 116-136.
- Simonton, D. 1999. "Creativity from a Historiometric Perspective", in *Handbook of Creativity* (see Sternberg, 1999).
- Sloboda, J. 1985/1999. *The Musical Mind (1st/2nd Edition)*. Oxford, UK: Oxford Science Publications.
- Sloboda, J. 2005. *Exploring the Musical Mind*. Oxford, UK: Oxford University Press.
- Smith, J., Mould, D., and Daley, M. 2009. "Constructures: supporting human ingenuity in software", in *Digital Creativity*, 20(1-2):79-94. UK: Routledge.
- Smith, B. and Smith, W. 1994 "Uncovering Cognitive Processes in Music Composition: Educational and Computational Approaches", in *Music Education: An Artificial Intelligence Approach* (ed. Smith, M., Smaill, A., and Wiggins, G.), pp.56-73. NY: Springer-Verlag.
- Smyth, M., Collins, A., Morris, P. and Levy, P. 1994. *Cognition in Action (2nd Edition)*. Hove, UK: Lawrence Erlbaum Associates.
- Sternberg, R. 1985. "Implicit theories of intelligence, creativity, and wisdom", in *Journal of Personality and Social Psychology*. 49: 607-627. Washington, DC: American Psychological Association.
- Sternberg, R. (ed.). 1999. *Handbook of Creativity*. Cambridge, UK: Cambridge University Press.
- Sternberg, R. 2003. *Wisdom, Intelligence, and Creativity Synthesized*. Cambridge, UK: Cambridge University Press.
- Sternberg, R. and Lubart, T. 1995. *Defying the crowd: Cultivating creativity in a culture of conformity*. New York: Free Press.
- Sternberg, R. and Lubart, T. 1999. "The Concept of Creativity: Prospects and Paradigms", in *Handbook of Creativity* (see Sternberg, 1999). pp. 3-15.
- Stowell, D., Robertson, A., Bryan-Kinns, N., and Plumbley, M. D. 2009. "Evaluation of live human-computer music-making: quantitative and qualitative approaches", in *International Journal of Human-Computer Studies*, 67:960-975. Elsevier Science.
- Swanwick, K. and Tillman, J. 1986. "The sequence of musical development: a study of children's composition", in *British Journal of Music Education*, 3:305-339.
- Sweetser, P. and Wyeth, P., 2005. "GameFlow: A Model for Evaluating Player Enjoyment in Games", in *ACM Computers in Entertainment*. 3(3):1-24.
- Tanimoto, S. 1990. "VIVA: A Visual Language for Image Processing", in *Journal of Visual Languages and Computing*. Academic Press. pp. 127-139.
- Tasajärvi, L. (ed.). 2004. *DEMOSCENE: the art of real-time*. Helsinki, Finland: EvenLakeStudios & katastro.fi.

- Thompson, S., and Lehmann, A. 2004. "Strategies for sight-reading and improvising music", in *Musical Excellence* (ed. Williamon, A). Oxford, UK: Oxford University Press. pp. 143-162.
- Torrance, E. 1962. *Guiding creative talent*. Englewood Cliffs, NJ: Prentice-Hall.
- Turkle, S. and Papert., S. 1992. "Epistemological Pluralism and the Revaluation of the Concrete", in *Journal of Mathematical Behavior*. 11(1):3-33.
- Venkatesh, V. 1999. "Creation of Favorable User Perceptions: Exploring the Role of Intrinsic Motivation", in *MIS Quarterly*. 23(2):239-260. University of Minnesota.
- Walker, M. 1999. "Mind the Gap: Dealing with Computer Audio Latency", in *Sound On Sound*, April 1999. Cambridge, UK: SOS Publications.
- Wallas, G. 1926. *The Art of Thought*. NY: Harcourt, Brace and World.
- Wang, C.K.J., Liu, W.C. and Khoo, A. 2009. "The Psychometric Properties of Dispositional Flow Scale-2 in Internet Gaming", in *Current Psychology*. 28(3):194-201. Springer.
- Ward, T., Smith, S., and Finke, R. 1999. "Creative Cognition", in *Handbook of Creativity* (see Sternberg, 1999). pp. 189-212.
- Webster, P. 1989. "Creative Thinking in Music: The Assessment Question", in *Suncoast Music Education Forum*. pp. 1-37. Available from: <http://www.eric.ed.gov> [Last Accessed: Dec 2011]
- Webster, P. 2002. "Creative Thinking in Music: Advancing a Model", in *Creativity and Music Education* (ed. Sullivan T. and Willingham L.). pp. 16-33. Edmonton: Canadian Music Educators Association.
- Wehner, L., Csikszentmihalyi, M., Magyari-Beck, I. 1991. "Current approaches used in studying creativity: An exploratory investigation", in *Creativity Research Journal*. 4(3):261-271.
- Weisberg, R. 1993. *Creativity: Beyond the Myth of Genius*. NY: Freeman.
- Weisberg, R. 1999. "Creativity and Knowledge: A Challenge to Theories", in *Handbook of Creativity* (see Sternberg, 1999). pp. 226-250.
- Weiser, M. 1999. "The Computer for the 21st Century", in *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(3):3-11.
- Wherry, M. 2009. "Digidesign Pro Tools 8", in *Sound On Sound*. Cambridge, UK. SOS Publications Group.
- White, P. 2000. *Desktop Digital Studio*. Sanctuary Publishing.
- White, 2001. *VST Instruments and VST Effects*. Sanctuary Publishing.
- Wiggins, G. and Pearce, M. 2001. "Aspects of a cognitive theory of creativity in musical composition", in *Proceedings of the ELA/02 Workshop on Creative Systems*, pp.17-24. Lyon, France.
- Williamon, A. 2004. *Musical Excellence: Strategies and techniques to enhance performance*. Oxford, UK: Oxford University Press.
- Wilson, F. 1998. *The Hand: How its use shapes the brain, language, and human culture*. New York: Pantheon.
- Winograd, T. and Flores, F. 1987. *Understanding Computers and Cognition: A New Foundation for Design*. Addison-Wesley.

This section provides a details of the pattern effects commands available in tracker programs, along with approximate equivalent MIDI functionality and common music notation. In addition to built-in effects; *IT2*, *reViSiT*, and *Renoise* also provide user-defined effects for directly controlling MIDI or plugins.

IT2 / reViSiT	Description	FT2	Renoise	MIDI	Musical Score ¹
Axx ²	Set resolution or speed to xx (fps).	Fxx (xx<20)	F1xx	-	<i>tempo direction</i>
Bxx	Jump to xx in order list.	Bxx	-	-	da    etc.
Cxx	Jump to xx row in the next pattern.	Dxx	FB00	-	(see above)
Dx0 D0x DFx DxF DEx DxE	Volume slide up/down with speed x (F denotes fine, E denotes extra fine)	Ax0 A0x EAx ExA - -	06xx 07xx - - - -	Key aftertouch	 <i>crescendo, diminuendo</i>
Exx EFx EEEx	Portamento down with speed xx (F denotes fine, E denotes extra fine)	2xx E2x X2x	02xx - -	Pitch bend up ⁴	
Fxx FFx FEx	Portamento up with speed xx (F denotes fine, E denotes extra fine).	1xx E1x X1x	01xy - -	Pitch bend down ⁴	
Gxx	Portamento from previous note to that specified in pitch column (speed xx).	3xx	05xx	-	
Hxy	Vibrato with speed x, depth y.	4xy	0Fxy	Pitch bend ⁴	<i>con vibrato</i>
Ixy	Mute after x frames, for y frames.	Txy	-	Key aftertouch	-
Jxy	Arpeggio (fast cycle of current pitch, and pitches at x and y semitones above).	0xy	00xy	Manual input	
Kxx	Repeat previous vibrato with volume slide (see Dxx).	6xx	-	Pitch bend ⁴ with key aftertouch	(see Hxy and Dxx)
Lxx	Repeat previous portamento with volume slide (see Dxx).	5xx	-	-	(see Hxy and Gxx)

IT2	Description	FT2	Renoise	MIDI	Musical Score ¹
Mxx	Set channel volume to xx.	Cxx	0Cxx	Channel volume ⁴	<i>f</i> , <i>p</i> , <i>mf</i> , etc.
Nx0	Channel volume slide up/down (F denotes fine)	-	-	Channel volume ⁴	 <i>crescendo, diminuendo</i>
N0x		-	-		
NFx		-	-		
NxF		-	-		
Oxx (w/ SAy)	Begin sample playback at offset yxx00h.	9xx (-)	09xx (-)	-	-
Px0	Panning slide left/right with speed x (F denotes fine)	P0x	9x	Pan position ⁴	-
P0x		Px0	Ax		
PFx		-	-		
PxF		-	-		
Qxy Q0y	Retrigger note every y frames, with volume macro x	Rxy E9y	0Exy 0E0y	Manual input (with velocity)	 (for particular instruments only)
Rxy	Tremolo with speed x, depth y.	7xy	-	Key aftertouch	
S3x	Set waveform for vibrato, tremolo and panbrello (sine, square, saw or random)	E4x	-	-	-
S4x		E7x	-	-	-
S5x		-	-	-	-
S70		-	-	-	<i>sostenuto, legato,</i>
S71	Previous note cut, off or fade	-	-	-	<i>staccato</i>
S72	-	-	-	-	-
S6x ³	Pattern delay (for x ticks)	EEx	FDxx	-	
S73	Set behaviour at note's termination (cut, continue, off, fade)	-	-	-	<i>sostenuto, legato,</i>
S74		-	-	-	<i>staccato</i>
S75		-	-	-	-
S76		-	-	-	-
S77	Enable / disable volume envelope	-	-	-	-
S78		-	-	-	-
S8x	Set channel pan / depth position	8xx	08xx	Pan position ⁴	-
S9x		-	0Axx		
SB0	Set start/end of repeated section (x times).	E60	-	-	
SBx		E6x	-	-	
SCx	Cut note (after x frames)	ECx	Fx (as volume)	Note off	
SDx	Delay note (for x frames)	EDx	0Dxx	Note on	n/a
Sex ³	Delay pattern (for x frames)	-	FDxx	-	

IT2	Description	FT2	Renoise	MIDI	Musical Score ¹
Txx ³	Set tempo to xx.	Fxx (xx>1F)	F0xx	Tempo change	
T1x ³ T0x ³	Tempo slide up/down (at speed x)	- -	- -	Tempo change	<i>accelerando, rallentando, ritardando</i>
Uxy	Fine vibrato with speed x, depth y	-	-	Pitch bend ⁴	<i>con vibrato poco</i>
Vxx	Set global volume to xx	Gxx	FCxx	-	<i>f, p, mf</i> , etc.
Wx0 W0x Wfx WxF	Global volume slide up/down at speed x (F denotes fine)	Hx0 H0x - -	- - - -	-	 <i>crescendo, diminuendo</i>
Xxx	Set panning position	8xx	08xx	Panning position ⁴	-
Wxy	'Panbrello' (panning oscillation) with speed x, depth y	-	-	Panning position ⁴	-
Zxx	Set filter cutoff / resonance	-	-	-	-
-	Rounds pitch to nearest semitone (glissando) if x is 1	E3x	-	-	-
-	Detunes note by x cents	E5x	-	-	-
-	Sets instrument envelope position	Lxx	-	-	-
-	'Volume Slicer' - ramps from current volume to x, to 0, after y ticks	-	04xy	Key aftertouch	
-	Set sample playback direction (backwards if xx = 00; forwards if 01)	-	0Bxx	-	-
-	Stop all notes and effects	-	FF00	MIDI reset	<i>silenzio</i>

¹ Approximate equivalences – may not hold in all situations.

² Note: unlike other trackers, *reViSiT* does not use resolution to set speed, but is instead slaved to the host. In *reViSiT*, Axx is used to control the number of subrows and granularity of effect playback.

³ Not supported in *reViSiT*, in order to maintain synchronisation with the host, which controls tempo.

⁴ Unlike trackers, MIDI command affects the entire *channel*, not individual notes.

The existence and role of *flow* (Csikszentmihalyi, 1990; see Section 3.7) in music has been widely established (Byrne *et al*, 2003; Chaffin and Limieux, 2004; MacDonald *et al*, 2006; Fritz and Avsec, 2007; Mullett, 2010). However, the study of flow experiences presents methodological challenges, including the disruptive influence of observation itself or the difficult subjects have reflecting on cognitive factors which they may not be conscious of (see Section 3.5). While the main dissertation looks at various ways to study the flow phenomenon using technology (i.e. using video, the internet, or interaction logging) to look at physical actions, user ability, and feedback; it is also possible to review historical literature on the experiences of composers (e.g. Graf, 1947; Boyd, 1992¹; Harvey, 1999), in which many of the more subjective qualities of flow are evident.

This appendix provides a collection of quotes from musicians, composers, and musicologists describing experiences corresponding to *flow* or one of its components. These accounts are presented here as evidence of flow in music composition (including notation use), but also as a reference for future *biographical* or *case study* approaches (see Policastro and Gardner, 1993) to studying flow in the context of musical creativity. Beyond general descriptions of flow-like experiences, quotes are organised by corresponding flow component (see Table 4, Section 3.7 for details).

General descriptions of Flow-like experiences

[I]t's free flow of information and inspiration, it's being in an altered state. It's very satisfying. Everything else disappears. It's like I'm part of a river and no matter what I did, I couldn't stop the current right then.

Rosanne Cash, songwriter
(Boyd, 1992; p163)²

I think you plug into this electricity—it's like a river in a way.

Peter Gabriel, producer
(Boyd, 1992; p170)²

Tchaikovsky has expressed well the concentration that is often required when creative activity seems to flow particularly well: "I forget everything and behave like a madman. Everything within me starts pulsing and quivering; hardly have I begun the sketch ere one thought follows another. In the midst of this magic process it frequently happens that some external interruption wakes me from my somnambulistic state: a ring at the bell, the entrance of my servant... Dreadful are such interruptions. Sometimes they break the thread of inspiration for a considerable time, so that I have to seek it again, often in vain" (From Newmarch 1906; reprinted in Vernon, 1970.)

(Sloboda, 1985; p137)

¹ Boyd (1992) is notable for its research into the "peak experience" (see Maslow, 1968) in music, and which can be seen as analogous to flow's "optimal experience" (Csikszentmihalyi, 1990).

² Musicians' uses of the word "flow", and reference to rivers and currents, mirror subject responses in Csikszentmihalyi's early interviews that directly led to the use of the word "flow" to describe the mental state (see Csikszentmihalyi, 2000).

Loss of Self-Consciousness / Ego

It's the time when you're not there that things happen. It's when your ego moves out of the way, when you create the space that music can actually come in and through. It's what we musician's live for, those magical moments.

Richard Thompson, songwriter
(Boyd, 1992; p162)

One feels oneself a transmitter; there is a loss of ego activity. There is a greater feeling of the unitive state where everything is possible; there is no individuation.

Jonathan Harvey, composer
(Deliege and Harvey, 2006; p31)

As the idea grows, you lose yourself.

Leoš Janáček, composer
(Harvey, 1999; p31)

While Haendel was composing, he isolated himself from the world. No visitor could get through to him; he locked himself in so that he could be alone with his ideas. He took notice of no one, and talked out loud to himself. He sobbed when moved by some text. Often the servant who brought Haendel his morning chocolate found him sobbing aloud, tears wetting the sheet of music he was writing on.

(Graf, 1947; p352)

Action-awareness merging

The world in which he ordinarily moves, and which is the scene of his activity, seems to vanish. He feels himself transplanted to another world wherein everything that would normally catch his interest ceases to exist: his work, his human relations, his worries and hopes and fears, his plans and his everyday sentiments.

(Graf, 1946; p3)

Many musicians referred to a kind of mental "stillness" that is necessary for the unconscious to make itself known through creative expression.

(Boyd, 1992; p84)

Numerous musicians described this exhilarating escape from normal consciousness during performance, which can feel like the music has taken on a life of its own.

(Boyd, 1992; p93)

I rely completely on instinct as a player. Often, while I'm playing, there are certain moments when I disappear.

Mick Fleetwood, guitarist and songwriter
(Boyd, 1992; p105)

Everything is in harmony both inside and out. It is the coming together of the conscious and the unconscious and one of the rare moments when the conscious mind is not fighting to keep the unconscious at bay.

(Boyd, 1992; p159)

*You get inside the music to such an extent that you kind of *are* the music, or the music's you. You're thinking about it but you're not thinking about it. Sometimes I think it's almost a flashing backwards and forwards of intellect and intuition.*

Richard Thompson, guitarist and songwriter
(Boyd, 1992; p162)

Once I've managed to transcend such things as where I am, who I'm playing with, how I'm playing, what the temperature of the room is, how the audience is, who the rest of the band is, that's when the real playing happens.

Ian Wallace, drummer
(Boyd, 1992; p171)

For many composers, this process of concentration requires them to cut themselves off completely from the everyday world, shutting the door firmly behind them: they become absorbed in the work in progress.

(Harvey, 1999; p32)

If it's an Allegro that pursues me, my pulse keeps beating faster, I can get no sleep. If it's an Adagio, the I notice my pulse beating slowly. My imagination plays on me as if I were a clavier.

Franz Josef Haydn
(Harvey, 1999; p32)

The composer frequently becomes so absorbed in the piece of music that it begins, for him, to constitute a separate, self-sufficient world. This is proved by the way in which composers write that they 'live in' or 'inhabit' their music: Beethoven wrote that 'I live entirely in my music', while Wagner wrote of Tristan that 'I am living wholly in this music ... I live in it eternally.'

(Harvey, 1999; p33)

Distorted Perception of Time

Time goes very quickly. It's like a suspension where one moment, it feels like it's only seconds long. But when I come out of that moment, it feels like it's only seconds long. I don't feel warm, I don't feel as if there's light around.

Graham Nash, songwriter
(Boyd, 1992; p85)

I had lost myself for four hours in this experience of painting... There were no distractions, there's just nothing else.

Rosanne Cash, songwriter (while painting)
(Boyd, 1992; p163)

An idea, when it arises, acknowledges neither spare time nor time that is tied down. It wakes you from sleep, slows or quickens your step during a walk ...

Leoš Janáček, composer
(Harvey, 1999; p31)

For Chopin, composition was such an all-consuming process that he lost any sense of the progress of time in the outside world: "How often I take night for day and day for night; how often I live in my dreams, and sleep in the daytime."

(Harvey, 1999; p33)

Focus & Concentration

By completely concentrating on the music they're playing or writing, musicians are able to open themselves up to a peak experience. It is as if an intense concentration can push the conscious mind away from "self-consciousness" and the unconscious is allowed to filter through.

(Boyd, 1992; p31)

Inspiration is a state of spirit, a state of mind, and - why not? - a state of ecstasy (in its rigorous sense of being carried away), in which all mental, psychic and spiritual forces of the individual concur intensely for a single purpose, that of creating, composing or investigating in a total concentration of faculties in a given direction. We do not call all cases of concentration inspiration, but all cases of inspiration involve concentration.

Carlos Chávez, composer
(Harvey, 1999)

Activity becomes Autotelic

I would try vainly to express in words that unbounded sense of bliss that comes over me when a new idea opens up within me and starts to take on definite form. Then I forget everything and behave like one demented. Everything inside me begins to pulse and quiver: I hardly being the sketch before one thought begins tumbling over another. There is something somnambulistic about this condition. "On ne s'entend pas vivre." It is impossible to describe such moments.

Pyotr Ilyich Tchaikovsky
(Harvey, 1999; p31)


[M]any musicians told me that the drive to create led to their immersing themselves completely in learning to play their instrument. What for some would have been hard work became a joy to these fledging artists. Maslow described this aspect of creative people's attitude toward working at their art form: "Duty [becomes] pleasure, and pleasure merge[s] with duty. The distinction between work and play [becomes] shadowy."

(Boyd, 1992; p71)

Haydn and Mozart composed a good deal of work in commission [...] But this music does not originate in a fantasy that dissolves all animating substances on its upward flow from the depths of the soul to its peak. [...] Even Beethoven complained, in 1823, that he was not writing what he would have liked to, but, "for the sake of money, that which I have to." [In such compositions], not all the smelting furnaces of his fantasy were working, not all cauldrons were lit, not all wheels were in motion.

(Graf, 1947; p79)

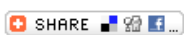
REGISTRATION FORM: This online survey was completed by all participants, joining the experiment. In addition to providing demographic information, questions were asked to establish levels of experience in music and relevant technology, used to correlate with interaction data. Appendix G provides an overview of responses given.






the reViSiT experiment

Be a part of the future of music technology... and get reViSiT Pro for free!

[home](#) [more info](#)
[register](#) [download](#)




External Links


Join the experiment!

Simply fill out the form and 3-minute questionnaire below, and your *reViSiT Pro* activation code will be emailed to you within moments. If you have any questions about the experiment or how your data is used, please take a look at the experiment's [frequently-asked questions](#).

All collected data is treated confidentially and anonymously.



*This research and its methods have been reviewed and **ethically-approved** by the University of Cambridge.*



UNIVERSITY OF CAMBRIDGE

LOGIN DETAILS

Email	<input type="text"/>	Enter the email address, to which you wish us to send your activation code for reViSiT Pro.
(confirm)	<input type="text"/>	
Password	<input type="password"/>	Choose a password to associate with your experiment account.
(confirm)	<input type="password"/>	

GENERAL INFORMATION

Gender	<input type="radio"/> Male <input type="radio"/> Female	This information is used to look for trends in our data that might correspond to user background.
Age Range	<input type="text" value=""/> ▾	
Location	<input type="text" value=""/> ▾	

EXPERIMENT QUESTIONNAIRE (2-3 minutes approx.)

The experiment looks at how skill develops, so we need to know a little about your general background and experience with computers and music technology, as well as any specific skills you might have picked up along the way.

How would you describe your experience and knowledge of computers in general?

- ☐ **Novice User** (e.g. new to computers)
- ☐ **Regular User** (e.g. email, internet, word processing)
- ☐ **Power User** (e.g. gaming, OS tweaking, upgrading)
- ☐ **Expert User** (e.g. coding, network admin)

How much experience have you had with trackers / tracking?

- ☐ 1 - I've never heard of them.
- ☐ 2 - I know what they are.
- ☐ 3 - I've played around with them a bit.
- ☐ 4 - I use them quite often.
- ☐ 5 - They form the foundation of my studio.

How much experience have you had with Impulse Tracker 2?

Please include any clones in you answer (e.g. ChibiTracker, zTracker, etc.), but exclude reViSiT.

- ☐ 1 - I've never heard of it.
- ☐ 2 - I know of it.
- ☐ 3 - I've played around with it a bit.
- ☐ 4 - I used to use it a lot.
- ☐ 5 - I still use it extensively.

Have you had any prior experience with reViSiT?

- ☐ 1 - I'd never heard of it.
- ☐ 2 - I knew of it.
- ☐ 3 - I've played around with it a bit.
- ☐ 4 - I've used it extensively.
- ☐ 5 - It forms the foundation of my studio.

What other music experience have you had?

Tick any that apply.

- | | |
|--|---|
| <input type="checkbox"/> I enjoy listening to music. | <input type="checkbox"/> I've had music lessons. |
| <input type="checkbox"/> I play the piano. | <input type="checkbox"/> I perform live. |
| <input type="checkbox"/> I play another acoustic musical instrument. | <input type="checkbox"/> I compose music/songs/tunes. |
| <input type="checkbox"/> I play several musical instruments. | <input type="checkbox"/> I am a professional performer. |
| <input type="checkbox"/> I can read music. | <input type="checkbox"/> I am a professional composer/songwriter. |

What music technology have you used in the past?

Tick any you've used, ticking twice if you've know it well.

Sequencers

- ☐ ☐ Ableton Live
- ☐ ☐ ACID
- ☐ ☐ Cakewalk/SONAR
- ☐ ☐ Cubase
- ☐ ☐ Digital Performer
- ☐ ☐ FL Studio
- ☐ ☐ Logic
- ☐ ☐ ProTools
- ☐ ☐ Reaper
- ☐ ☐ Rosegarden
- ☐ ☐ Tracktion
- ☐ ☐ Other:

Notation Software

- ☐ ☐ Encore
- ☐ ☐ Finale
- ☐ ☐ Lilypond
- ☐ ☐ Sibelius
- ☐ ☐ Other:

Trackers

- ☐ ☐ Amiga Trackers
- ☐ ☐ Early PC/DOS Trackers
- ☐ ☐ Buzz
- ☐ ☐ Fast Tracker 2
- ☐ ☐ Impulse Tracker 2
- ☐ ☐ ModPlug Tracker
- ☐ ☐ MadTracker
- ☐ ☐ OctaMED (PC)
- ☐ ☐ Renoise
- ☐ ☐ reViSiT
- ☐ ☐ Scream Tracker 2/3
- ☐ ☐ Other:

Other Software

- ☐ ☐ Band in a Box
- ☐ ☐ CSound
- ☐ ☐ energyXT
- ☐ ☐ Final Scratch
- ☐ ☐ GarageBand

Audio Editors

- ☐ ☐ Audacity
- ☐ ☐ Audition
- ☐ ☐ Sound Forge
- ☐ ☐ WaveLab
- ☐ ☐ Other:

Hardware

- ☐ ☐ Analogue/Digital Synths
- ☐ ☐ Control Surfaces
- ☐ ☐ Hardware Samplers
- ☐ ☐ Hardware Sequencers
- ☐ ☐ MIDI Keyboards
- ☐ ☐ Other MIDI Controllers
- ☐ ☐ Microphones
- ☐ ☐ Mixing Consoles
- ☐ ☐ Outboard Effects
- ☐ ☐ Recording Equipment
- ☐ ☐ Live/PA Equipment
- ☐ ☐ Other:

Which descriptions describe how you interact with the computer?

Tick any that apply.

Mouse

- ☐ I prefer to use the mouse.
- ☐ I use the mouse a lot.
- ☐ I often use the mouse scroll wheel.
- ☐ I avoid using the mouse.
- ☐ I spend most of my time using the mouse.

Keyboard

- ☐ I prefer to use the keyboard.
- ☐ I use the keyboard a lot.
- ☐ I can type quickly.
- ☐ I can type touch-type.
- ☐ I try to use keyboard shortcuts.
- ☐ I spend most of my time using the keyboard.

Other

- | | |
|---|---|
| <input type="checkbox"/> I tend to enter music by audio/microphone. | <input type="checkbox"/> I tend to enter music by MIDI. |
| <input type="checkbox"/> I like software to be simple and easy. | <input type="checkbox"/> I like software to be powerful and advanced. |
| <input type="checkbox"/> I find computers a chore. | <input type="checkbox"/> I spend a lot of time on computers. |
| <input type="checkbox"/> I enjoy using computers. | <input type="checkbox"/> I suffer discomfort using the computer (e.g. RSI). |

INFORMED CONSENT STATEMENT

I have been informed about the purpose and nature of this experiment, and I have agreed to take part. I understand that taking part in this experiment is voluntary and I can pull out of the experiment at any time.

I understand that data collected during my use of this program will be stored on a computer and may contribute to scientific publications and presentations. I agree that the data can be made available anonymously for other researchers, inside and outside the Computer Laboratory. This data will not be linked to me as an individual.

- ☐ I agree to the above terms and conditions.

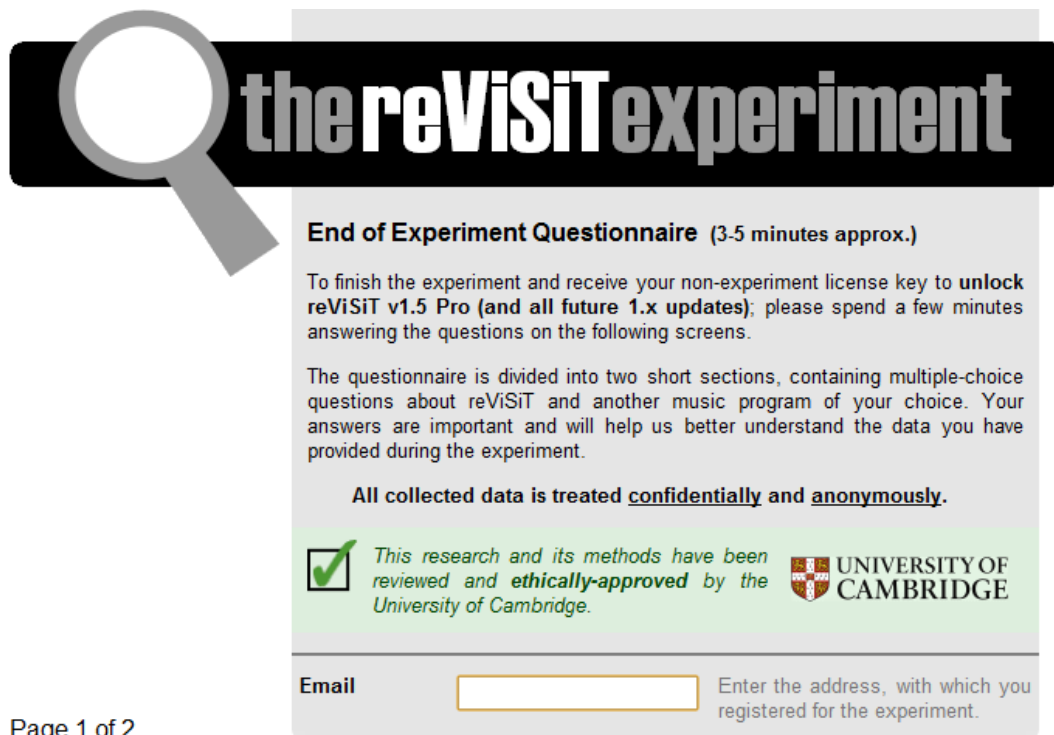
COPYRIGHT and DISCLAIMER

This software is for use by the participants of the reViSiT Experiment. Any modification to, or reselling of, any part of the software is strictly prohibited without explicit permission of the developer. Neither Chris Nash (the developer) nor the University of Cambridge accept any responsibility for undesirable effects (including data loss, etc.) arising from the use or misuse of the software. Although the software has been rigorously tested, the software is provided AS IS.

- ☐ I agree to the above terms and conditions.

[Register for the experiment!](#)

END-OF-EXPERIMENT QUESTIONNAIRE: This survey repeats questions from the registration, for comparison with collected interaction data, and to support before/after comparisons, to see how attitudes to *reViSiT* changed. The second online page also surveyed how individuals were using their host sequencer, to help interpretation of the limited host interaction data. However, the main objective of the survey was to analyse flow in notation use using the *Cognitive Dimension of Notations* framework (Green and Petre, 1996). To assess flow, two batteries of nine questions are presented, corresponding to statements about the nine components of flow. These statements are derived from the *dispositional flow scale (DFS)-2* (Jackson and Eklund, 2002), a psychometric test that uses four batteries of nine questions, reduced here to keep the survey short and accommodate additional questions on cognitive dimensions. In the latter case, a single battery of 16 questions is presented, corresponding to 15 standard cognitive dimensions, adapted from the *Cognitive Dimensions Questionnaire Optimised for Users* (Blackwell and Green, 2000), and one additional statement corresponding to virtuosity (“With time, I think I could become a virtuoso user of the system.”). In each case, statements are scored on a 5-point Likert scale (Strongly Disagree-Strongly Agree), the results from which are analysed in Chapter 9.



the reViSiT experiment


End of Experiment Questionnaire (3-5 minutes approx.)

To finish the experiment and receive your non-experiment license key to **unlock reViSiT v1.5 Pro (and all future 1.x updates)**; please spend a few minutes answering the questions on the following screens.

The questionnaire is divided into two short sections, containing multiple-choice questions about reViSiT and another music program of your choice. Your answers are important and will help us better understand the data you have provided during the experiment.

All collected data is treated **confidentially** and **anonymously**.

☒ This research and its methods have been reviewed and **ethically-approved** by the University of Cambridge.

 **UNIVERSITY OF CAMBRIDGE**

Email Enter the address, with which you registered for the experiment.

Page 1 of 2

Questions about reViSiT

This first section recalls 3 questions from the original questionnaire, when you signed up, regarding your experience with trackers and reViSiT. In each case, please answer these questions again, to reflect any changes that might have happened in the time since.

1. How much experience have you now had with trackers / tracking?

- ☐ 1 - I'm still not sure what they are.
- ☐ 2 - I know what they are.
- ☐ 3 - I've played around with them a bit.
- ☐ 4 - I use them quite often.
- ☐ 5 - They form the foundation of my studio.

2. How much experience have you now had with reViSiT?

- ☐ 1 - I've never heard of it.
- ☐ 2 - I know of it.
- ☐ 3 - I've played around with it a bit.
- ☐ 4 - I've used it extensively.
- ☐ 5 - It forms the foundation of my studio.

3. Which descriptions describe how you interact with reViSiT?

Tick any that apply.

Mouse

- ☐ I prefer to use the mouse.
- ☐ I use the mouse a lot.
- ☐ I often use the mouse scroll wheel.
- ☐ I avoid using the mouse.
- ☐ I spend most of my time using the mouse.

Keyboard

- ☐ I prefer to use the keyboard.
- ☐ I use the keyboard a lot.
- ☐ I can type quickly.
- ☐ I can type touch-type.
- ☐ I try to use keyboard shortcuts.
- ☐ I spend most of my time using the keyboard.

Other

- ☐ I enjoy using reViSiT.
- ☐ I find reViSiT simple and easy.
- ☐ I find reViSiT a chore.
- ☐ I normally have both hands on the keyboard.
- ☐ I normally have one hand on the keyboard and one hand on the mouse.
- ☐ I tend to enter music by MIDI.
- ☐ I find reViSiT powerful and advanced.
- ☐ I spend a lot of time using reViSiT.
- ☐ I suffer discomfort using reViSiT (e.g. RSI).

4. About the reViSiT interface and notation...

Please answer the following questions in relation to your use of the program, during the experiment. There are no right or wrong answers. Think about how you felt while using the program and answer the questions using the rating scale.

Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	1. It is easy to view and find parts of the music during editing.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	2. It is easy to compare different parts of the music.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	3. When writing music with the notation, there are difficult things I have to work out in my head.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	4. It is easy to go back and make changes to the music.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	5. The notation is concise and makes good use of space.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	6. Within the overall scheme of my music, it is easy to see what each part is for.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	7. It is easy to make annoying mistakes.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	8. The notation matches how I would describe the music myself.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	9. I can sketch things out and play around with ideas, without having to be too precise about the exact result.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	10. It is hard to see how different parts of the music relate to each other.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	11. It is easy to stop and check my work while creating or editing it.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	12. Where aspects of the notation mean similar things, the similarity is clear in the way they appear.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	13. I can edit the song in any order, without having to think or plan ahead.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	14. I'm able to make informal notes to myself that aren't part of the music.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	15. I do things to customise, adapt or use the program in ways its designer may not have intended.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	16. With time, I think I could become a virtuoso user of the system.

5. About your reViSiT user experience...

Please answer the following questions in relation to your experiences in using the program during the experiment. These questions relate to the thoughts and feelings you may have experienced while taking part. There are no right or wrong answers. Think about how you felt during the activity and answer the questions using the rating scale, left.

Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	1. I was challenged, but my skills allowed me to meet the challenge.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	2. I made the correct movements without thinking about trying to do so.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	3. I knew clearly what I wanted to do.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	4. I was aware of how well I was performing.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	5. It was no effort to keep my mind on what was happening.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	6. I had a sense of control over what I was doing.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	7. I was not concerned with how others may evaluate my performance.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	8. Time seemed to alter (either slowed down or speeded up).
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	9. I really enjoyed the experience.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	10. The challenge and my skills were at an equally high level.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	11. I did things spontaneously and automatically without having to think.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	12. My goals were clearly defined.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	13. I could tell by the way I was performing how well I was doing.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	14. I was completely focused on the task at hand.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	15. I had a feeling of total control.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	16. I was not worried about what others may have been thinking of me.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	17. I lost my normal awareness of time.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	18. I found the experience extremely rewarding.

Page 2 of 2

Questions about your sequencer / DAW

Please now select a different music program to answer questions on. This can be the program you used to host reViSiT or another software package, but not another tracker.

- | | | |
|---|--|---|
| <input type="radio"/> Ableton Live | <input type="radio"/> Apple / Emagic Logic | <input type="radio"/> FL Studio |
| <input type="radio"/> Sony Acid | <input type="radio"/> ProTools | <input type="radio"/> Presonus Studio One |
| <input type="radio"/> Cakewalk SONAR | <input type="radio"/> Reaper | |
| <input type="radio"/> Steinberg Cubase / Nuendo | <input type="radio"/> Rosegarden | |
| <input type="radio"/> Digital Performer | <input type="radio"/> Mackie Traktion | <input type="radio"/> Other: <input type="text"/> |

Version:

How much experience have you had with the program?

- ☐ 1 - I've never heard of it.
☐ 2 - I know of it.
☐ 3 - I've played around with it a bit.
☐ 4 - I've used it extensively.
☐ 5 - It forms the foundation of my studio.

Where do you spend most time in the program?

- | | | |
|---|--|---|
| <input type="radio"/> Arrange Window / Track View | <input type="radio"/> Step Sequencer | <input type="radio"/> Mixer |
| <input type="radio"/> Score Editor | <input type="radio"/> Transport Bar | <input type="radio"/> Effects or Plugins |
| <input type="radio"/> Piano Roll | <input type="radio"/> Audio Editor | <input type="radio"/> Settings (e.g. automation) |
| <input type="radio"/> Data List | <input type="radio"/> Modular Synth / Routing View | <input type="radio"/> Other: <input type="text"/> |

Which descriptions describe how you interact with the program?

Tick any that apply.

Mouse

- ☐ I prefer to use the mouse.
- ☐ I use the mouse a lot.
- ☐ I often use the mouse scroll wheel.
- ☐ I avoid using the mouse.
- ☐ I spend most of my time using the mouse.
- ☐ I normally have one hand on the keyboard and one hand on the mouse.

Keyboard

- ☐ I prefer to use the keyboard.
- ☐ I use the keyboard a lot.
- ☐ I try to use keyboard shortcuts.
- ☐ I spend most of my time using the keyboard.
- ☐ I normally have both hands on the keyboard.

Other

- | | |
|--|--|
| <input type="checkbox"/> I tend to use a mic to enter music. | <input type="checkbox"/> I tend to enter music by MIDI. |
| <input type="checkbox"/> I use a hardware MIDI controller. | <input type="checkbox"/> I tend to record acoustic musical instruments. |
| <input type="checkbox"/> I use a hardware control surface or digital mixer to control the program. | |
| <input type="checkbox"/> I enjoy using the program. | <input type="checkbox"/> I find the program powerful and advanced. |
| <input type="checkbox"/> I find the program simple and easy. | <input type="checkbox"/> I spend a lot of time using the program. |
| <input type="checkbox"/> I find the program a chore. | <input type="checkbox"/> I suffer discomfort using the program (e.g. RSI). |

About the program's interface and notation...

Please answer the following questions in relation to your use of the program, during the experiment. There are no right or wrong answers. Think about how you felt while using the program and answer the questions using the rating scale.

Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	1. It is easy to view and find parts of the music during editing.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	2. It is easy to compare different parts of the music.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	3. When writing music with the notation, there are difficult things I have to work out in my head.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	4. It is easy to go back and make changes to the music.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	5. The notation is concise and makes good use of space.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	6. Within the overall scheme of my music, it is easy to see what each part is for.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	7. It is easy to make annoying mistakes.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	8. The notation matches how I would describe the music myself.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	9. I can sketch things out and play around with ideas, without having to be too precise about the exact result.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	10. It is hard to see how different parts of the music relate to each other.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	11. It is easy to stop and check my work while creating or editing it.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	12. Where aspects of the notation mean similar things, the similarity is clear in the way they appear.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	13. I can edit the song in any order, without having to think or plan ahead.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	14. I'm able to make informal notes to myself that aren't part of the music.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	15. I do things to customise, adapt or use the program in ways its designer may not have intended.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	16. With time, I think I could become a virtuoso user of the system.

About your program's user experience...

Please answer the following questions in relation to your experiences in using the program during the experiment. These questions relate to the thoughts and feelings you may have experienced while taking part. There are no right or wrong answers. Think about how you felt during the activity and answer the questions using the rating scale, left.

Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	1. I was challenged, but my skills allowed me to meet the challenge.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	2. I made the correct movements without thinking about trying to do so.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	3. I knew clearly what I wanted to do.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	4. I was aware of how well I was performing.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	5. It was no effort to keep my mind on what was happening.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	6. I had a sense of control over what I was doing.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	7. I was not concerned with how others may evaluate my performance.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	8. Time seemed to alter (either slowed down or speeded up).
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	9. I really enjoyed the experience.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	10. The challenge and my skills were at an equally high level.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	11. I did things spontaneously and automatically without having to think.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	12. My goals were clearly defined.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	13. I could tell by the way I was performing how well I was doing.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	14. I was completely focused on the task at hand.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	15. I had a feeling of total control.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	16. I was not worried about what others may have been thinking of me.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	17. I lost my normal awareness of time.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	18. I found the experience extremely rewarding.

This appendix contains descriptions of the different events collected from users by *iMPULS* (see Chapter 5). For each event, a summary of the encoding, decoding and use is provided. All encoded numeric values are given in hexadecimal, other hex values are denoted with an ‘h’ suffix.

Command (a), 4 bytes.

A program command that has been triggered, identified by enumerator and accompanied with a single integer or floating-point parameter to provide additional information, depending on context. Identifies many one-shot user actions, covering file, playback, clipboard and other editing operations. Later used to explicitly identify commands triggered by user-remapped keyboard input (v1.1+) and operations triggered by mouse interaction (v1.3+).

		<i>Examples</i>
a 12 78	CMD_HOSTPLAY (12h), 120 (78h)	⇒ <i>Host playback at 120bpm</i>
a 22 03	CMD_KEYBOARD (22h), kLoadSong (03h)	⇒ <i>Keyboard shortcut to Load Song</i>
a 25 20	CMD_MOUSE (25h), kOverwrite (20h)	⇒ <i>Mouse operation to Drag Copy (Overwrite)</i>

File (F), 669,203 bytes (compresses to ~1 kilobyte).

A summary of the musical content, related to a loaded/saved file. To protect the privacy of the user’s data, the content of their files is not collected. Instead, each time a song is loaded or saved, only a brief summary of specific aspects of the music is recorded, insofar as they relate to how the program is used – the overall distribution of pitches and effects entered, the areas of the pattern used and the parts of the music auditioned. Data is maintained as a static array of integer counts – incremented, for example, when the user listens to a specific pattern. The resulting dataset is sparse (mostly 0’s), and thus responds favourably to the subsequent (ZIP) compression of the log (~600:1). During collection, the static, uncompressed structure (see below) is maintained to avoid dynamic memory allocations or on-the-fly compression, which would impact program (audio) performance:

```

UINT patterns_total; // total patterns used in song

// for each cell (track, row), number of patterns containing data for...
UINT pitch_coverage[MAX_TRACKS][MAX_ROWS]; // ... pitch
UINT instrument_coverage[MAX_TRACKS][MAX_ROWS]; // ... instrument
UINT volume_coverage[MAX_TRACKS][MAX_ROWS]; // ... volume
UINT panning_coverage[MAX_TRACKS][MAX_ROWS]; // ... panning
UINT depth_coverage[MAX_TRACKS][MAX_ROWS]; // ... depth
UINT effectcommand_coverage[MAX_TRACKS][MAX_ROWS]; // ... effect command
UINT effectparam_coverage[MAX_TRACKS][MAX_ROWS]; // ... effect parameter

// overall distribution of...
UINT pitch_usage[MAX_INSTRUMENTS][MAX_PITCHES]; // ... notes used (by inst./pitch)
UINT instrument_usage[MAX_INSTRUMENTS]; // ... instruments used
UINT volume_usage[MAX_VOLUMES]; // ... volumes used
UINT panning_usage[MAX_PANNINGS][MAX_PANNINGS]; // ... panning/depth used
UINT effect_usage[MAX_EFFECT_COMMANDS][256]; // ... effect/params used

UINT pattern_plays[MAX_PATTERNS]; // ... patterns played
UINT row_plays[MAX_TRACKS][MAX_ROWS]; // ... pattern cells played
UINT instrument_plays[MAX_INSTRUMENTS]; // ... instruments played
UINT effect_plays[MAX_EFFECT_COMMANDS][256]; // ... effect/params played

```

Focus (f), 3 bytes.

The current keyboard focus, specifying enumerated values for page, tab and control. Sets the context for future **Data** entries.

		<i>Examples</i>
f 01 00 00	PATTERN_EDITOR (01), PATTERN (00)	⇒ <i>Pattern Editor, Pattern</i>
f 04 02 05	INSTRUMENT_LIST (04h), INSTR_PANNING (02h), ILUI_LOOPBEGIN (05h)	⇒ <i>Instrument List, Panning Tab, Envelope Loop Begin (edit box)</i>

Cursor (c), 8 bytes.

The position of the keyboard cursor and offset of the viewport, within the current area of the program (see **Focus**). For the Pattern Editor, the original Cartesian coordinates (x,y – column, row) later extended to include subrow cursor position, in high-definition patterns (x₁|x₂,y₁|y₂ – track|column, row|subrow), splitting the 2 x 2-byte subscripts into 4 x 1-byte subscripts.

Examples

```
c 000C 000E 0000 000A      1(0Ch=12, 12/9), 3(12 mod 9), 14(0Eh), (00h=0, 0Ah=10)
                               ⇒ Track 2, Volume column, Row 14, Offset (0 tracks, 10 rows)
c 0103 010E 0000 000A      1(01h), 3(03h), 1(01h), 14(0Eh), (00h=0, 0Ah=10)
                               ⇒ As above, but Subrow 1
```

Help (s), up to 66 bytes.

Logs activity relating to the support systems, including context-sensitive popups and Windows Help usage. When the Windows help system is called, a callback function is specified to receive updates of user activity from the separate Windows HTMLHelp process. These callbacks detail what user actions are taken (e.g. buttons pressed) and which help pages are viewed. Pages are identified using a local URL, to pages embedded in the help file.

Examples

```
s 01 02      HELP_TRACK(01), HHACT_TAB_SEARCH(02)
                               ⇒ HTML Help 'Search' tab selected
s 02 18 "reViSiT.chm::Credits.htm"  HELP_NAVIGATE(02), "reViSiT.chm::Credits.htm"
                               ⇒ reViSiT Credits page displayed
```

Keyboard (k), 6 bytes.

A single keyboard event, including details about the context, such as modifiers (Shift, Ctrl, Alt), up/down status, repeat count and associated final character. Note: this entry only details the key pressed, not the command that was triggered (see *CMD_KEYBOARD*, under **Command**).

Examples

```
k 61 01 00 41 00 02      'a'(61h), Shift(01), VKEY_NA(00), 'A'(41h), DOWN, 2
                               ⇒ Shift-a ('A') pressed (2nd repeat)
k 00 0A 16 00 01 00      "(00), Ctrl|Alt(08|02), VKEY_DELETE(16h), "(00), UP, 0
                               ⇒ Ctrl-Alt-Delete released
```

Mouse (m), 5 bytes.

A single mouse event, detailing the pointer location and buttons (or modifier keys) depressed. In *reViSiT*, the object clicked can be identified using either the previous **Focus** entry or, in the event the mouse is used to change focus, the **Focus** entry immediately subsequent. If the mouse is used to change a value, the value change will appear as a subsequent **Data** entry.

Example

```
m 0064 00C4 01      (0064h=100, 00C4=200), kLeft(01h)
                               ⇒ Left click at (100,200)
```

Notification (n/h), 12 bytes.

A Windows notification message. These messages represent the primary method of inter-process communication, in Windows. As such, they are always associated with a *window handle (HWND)* and contain details of operations concerning that window, including user input (e.g. keyboard, mouse), window operations (e.g. creating, moving, sizing, gaining focus) and many other context-sensitive functions (e.g. scrolling, user-defined messages, timers). *reViSiT*'s user input is recorded through its built-in keyboard and mouse handlers (see **Keyboard** and **Mouse**), so the main benefit of logging this data is to inspect window-related activity (moving, sizing, etc.). *reViSiT* also has access to the same information for the host application – through a *windows hook*, which it uses to ensure it gets keyboard input. This allows us to study an aspect of the host application – its keyboard, mouse and window activity. Note, however, that this is simply the raw input, and contains little information about what specific keys, clicks or windows do in the program. As such, notification entries are identified with a **n** for *reViSiT* notifications, and an **h** for host (or hooked) notifications. The window associated with any given message is identified using the last **Window** entry.

Examples

```
n 00000003 00000000 006400C4      WM_MOVE(0003h), (64h=100, C4h=200)
                               ⇒ Move reViSiT window to (100,200)
h 00000201 00000001 01200045      WM_LBUTTONDOWN(0201h), MK_LBUTTON(01h), (120h=288, 45h=69)
                               ⇒ Left Mouse Click at (288, 69) in host window
```

Window Information (W), up to 546 bytes.

Summary details of a window object, associated with a given *window handle (HWND)* – including position, size, window class, window styles and relation to other windows. In Windows, the central role of window objects in both inter-process communication and user input means that developers often use them in roles that don't correspond to distinct UI objects, as perceived to the user. To identify which are actually involved in the interaction, we inspect the *window style*, which contains bit flags used to change the appearance or behaviour of a window, e.g. `WS_VISIBLE` (window is visible), `WS_MAXIMIZE` (window is maximised), `WS_CHILD` (window appears inside the parent window). There is little information about the role of the window, but certain standard controls and dialogs can be recognised from the window's class name (e.g. `SCROLLBAR`) – increasingly, however, applications customise (“skin”) their interface and use custom UI toolkits with non-standard class names.

Example

```
w 04004325 04004320 05 "Mixer" 17 "SteinbergWndClass" 64 64 C4 C4 50000000 00000000
                                04004325h,04004320h,"Mixer","SteinbergWndClass",
                                (64h=100,64h=100,C4h=200,C4h=200),WS_CHILD(50000000h)|WS_VISIBLE(10000000h),00000000
                                ⇒ Cubase child window ("Mixer") at (100,100,200,200), currently visible
```

Window (w), 4 bytes.

Signals a change in the window context. This specifies the window associated with subsequent **Notification** entries. It does not indicate the window focus, on the user's desktop (which is extracted from the **Notification** entries themselves – i.e. `WM_SETFOCUS`). Instead, the entry is simply used to avoid having to include the *window handle (HWND)* in the **Notification** entries, which becomes redundant for flurries of activity within a single window. The entry simply records the window handle, which can be used to look-up more detailed information in the corresponding **Window Info** entry. During logging, if the window has not previously been seen, a **Window Info** entry is automatically created and added to the log, before the **Window** entry.

Example

```
w 04004325                                ⇒ Subsequent Notification entries relate to window with handle 04004325h
```

Version (v), up to 268 bytes.

Entered as the first entry of any log file, this entry records the manufacturer name, product name and version number of both the plugin and host.

Example

```
v 09 "Steinberg" 0A "Cubase VST" 00001FA4          "Steinberg","Cubase VST",8100(1FA4h)
  07 "nashNET" 07 "reViSiT" 000003EA          "nashNET","reViSiT",1002(03EAh)
                                              ⇒ Steinberg Cubase SX3, nashNET reViSiT 1.00.2
```

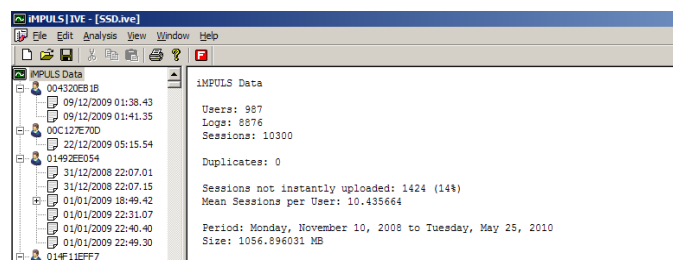
This section contains images and descriptions of the visualisations supported by the *iMPULS|IVE* program, used both to maintain the experiment over its 2-year run, and to explore and test the models and analysis methods detailed in Chapters 7 to 9.

Data Summaries

These visualisations were used to monitor the uptake of both the *reViSiT* program and experiment, providing a summary of data collected, broken down by day, week or user.

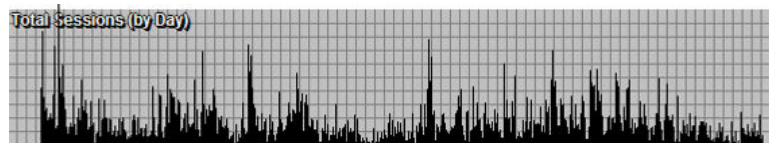
Overall Summary

The root node of the data hierarchy, labelled “*iMPULS* Data”, presents a screen where summary information about the experiment and whole corpus of data is display. It is designed to indicate the progress of the experiment, during its execution, and summarise the amount of data that has been collected. However, it can also be used to display visualisations or summaries of data across all users and sessions (e.g. **Distraction Chart**, **Integrity Check**).



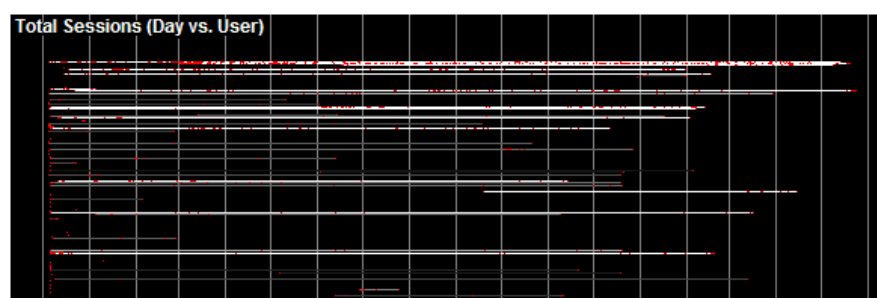
Total Session (by Day)

Used to track uptake and user activity during the experiment period, plotting the number of sessions for each day of the experiment, since 1 December 2008. The graph shows small spikes on weekends (when users have more time for *reViSiT*) and larger spikes around new *reViSiT* version releases.



Total Sessions (Day vs. User)

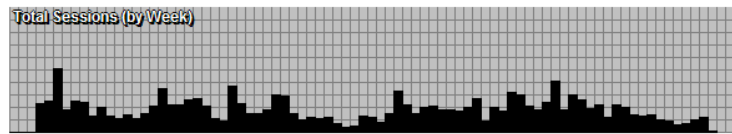
Used to track uptake, and user activity, as well as the amount of data for each user and how their activity is distributed over the duration of the experiment, plotting time (in days, across) against user (down). Each horizontal line represents a user's presence in the experiment –the more sessions they have contributed, the brighter the line (dark grey to white), with individual session marked on the timeline in red. Users are ordered by the order they registered with the experiment, though there can be a delay between the time of registration and the successful first submission of session data (e.g. installation problems, loading, distractions, loss of interest). The dense concentration of red-speckled, brighter lines at the beginning represents some of the more frequent users, who have been following the *reViSiT* project since before the experiment.



Total Sessions (by Week)

Used to track uptake and user activity during the experiment period, plotting the average number of

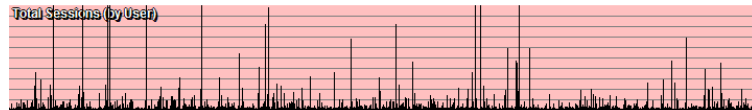
sessions for each week of the experiment, since 1 December 2008. Eliminates the effect of weekend spikes, giving a more useful indication of how much the program is used, though still showing spikes for weeks containing new releases.



Total Sessions (by User)

Plots the number of sessions for each user (ordered by User ID) in

the experiment, to get an idea of how many users have kept with the programme, and provide data for longer experiences, which might show behavioural changes and development. Sessions can potentially be very short (for example, when the loading problems are encountered), so plotting interaction duration may be more useful.



Interaction Visualisations

These visualisations present the interaction data from users and sessions, either aggregated or individually, and were used to provide broad overviews, based on filtered data or abstract models, as well as close-in detail, such as the original logs.

User Summary

The primary child nodes in the data hierarchy represent the individual users in the experiment. By default, all users are displayed, but the View menu can be used to restrict those displayed to users who claim specific levels of experience, or eliminate users that have provided insufficient data.

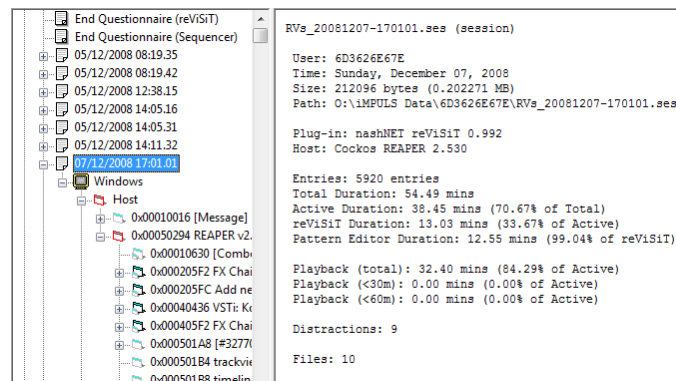
For each user, a summary view is available, displaying a variety of details concerning the user's registration, and the

interaction data submitted. This includes their responses to the initial experiment questionnaire, including their stated previous experience with computers, trackers, IT2 (a tracker) and reViSiT itself, as well as their skills with music, particular music programs and various interaction preferences (e.g. input device preferences).

A screenshot of the IMPULSIVE - [SSD.exe] application window. The window is divided into two main panes. The left pane shows a tree view of user data, with the user '004320EB1B' selected. The right pane displays the summary information for this user. The summary includes: ID: 1564 (user), Code: 004320EB1B (hash: 030BB / id: E2140), Joined: 10/11/2009, IP: 195.245.179.94, Number of logs: 2, Number of sessions: 2, Size of sessions: 0.036955 MB, Email Address: koshduksi@gmail.com, Gender: Male, Age: 40-49, Location: Portugal. It also lists General Experience (Computer: 000, Tracker: 000, IT2: 000, reViSiT: 000), Music Experience (Listener: (I enjoy listening to music.), Literate: (I can read music.), Lessons: (I've had music lessons.), Performer - Piano: (I play the piano.), Composer: (I compose music/songs/tunes.)), Music Technology Experience (Sequencers: Expert: 2 - Other Sequencer, Reason; Used: 1 - Ableton Live; Trackers: Expert: 2 - Impulse Tracker, Screem Tracker; Used: 5 - Early/DOS, Fast Tracker, ModTracker, ModPlug Tracker, reViSiT; Hardware: Expert: 1 - Keyboards; Used: 0), Overall (Expert: 5, Used: 11), and Interaction Preferences (I use the mouse a lot., I often use the mouse scroll wheel., I use the keyboard a lot., I can type quickly., I try to use keyboard shortcuts., I tend to enter music by MIDI., I like software to be simple and easy., I enjoy using computers., I spend a lot of time on computers., I suffer discomfort using the computer.).

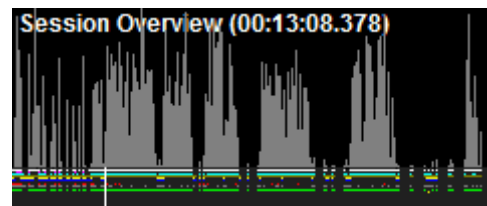
Session Summary

As with users, sessions present a summary of the data contained, including time and date, host and plugins used, the duration of the session, the total percentage of time with music playing, and the number of files opened or saved. The left hierarchy allows the user to drill down to individual windows and file summaries.

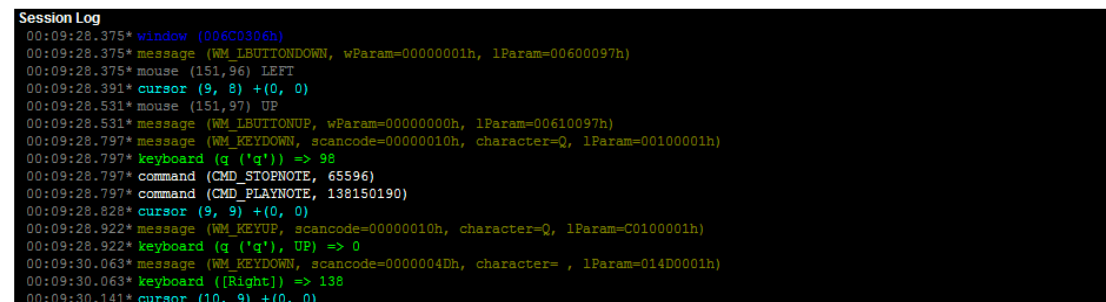


Session Overview

Displays a timeline of the session, providing an overview of its contents, marking events on a linear timeline. An area is reserved above the timeline for graphing filtered events, specific metrics or activity (pictured showing keyboard activity). The timeline uses the same colour-coding as the **Session (and User) Log**, and is used as a scrub bar, showing a cursor used to offset the starting time of the log data in such other views. The current cursor time is shown in brackets, beside the visualisation's caption.



Session Log



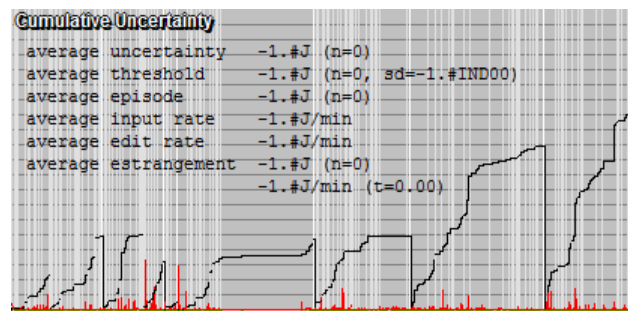
Displays the contents of a user's interaction log(s). Entries are colour-coded, depending on type, and can be filtered to include (or exclude) specific entries (see below), using the *iMPULS|IVE* filter system. The **Session Overview** visualisation (above) is used to control the starting time offset of the entries to be displayed, else the log can be browsed using the scroll bar. An abbreviated form of the log can also



be shown (left), with each entry as a single letter, spaced to indicate approx. temporal relationship to neighbouring events. This provides a concise overview of the log (e.g. on one page), and allows episodes of high activity, or breaks in interaction, to be quickly identified visually.

Uncertainty Graph

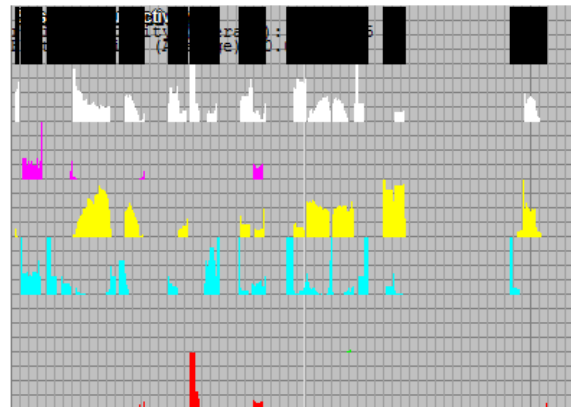
Charts the level of ‘uncertainty’ at any given moment (or event), during interaction (in a session, or across all the user’s sessions). Uncertainty (black) is calculated as the number of data edits made before the resulting music is auditioned – small edits (e.g. note entry) are differentiated from more involved edits (e.g. selection-level



edits, such as copy-‘n’-paste, or other selection editing), which trigger a greater level of uncertainty. Uncertainty inversely correlates to liveness, and thus should be minimised to support greater liveness, where domain feedback should be as frequent and immediate as possible. The ensuing analyses also collected information about editing episodes, and displays summary statistics on the graph. Additionally, the red line represents the number of edits/input events. Breaks in interaction are also indicated – light grey for distractions (see Distraction Analysis), white for end of session. The time axis can be linear, by time (scaled to fit), or ordinal, by event.

Context Chart

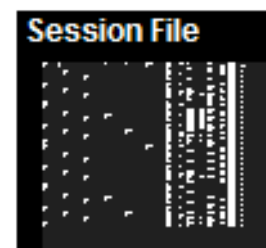
Visualises the contexts of keyboard input, within a log – see **Context Analysis**. Input events are bucketed into a set interval and their frequency over time is plotted for each context (colour-coded), as well as overall (black). Breaks in interaction (session ends, distractions) are marked with a vertical grey line. The visualisation has two modes – absolute and relative (pictured). The first simply plots the frequency of events, for any one interval;



the frequency relative to total input during that interval – indicating periods of interaction that are more, or less, characterised by specific contexts. As the images illustrate, however, the relative mode does not account for periods when interaction is overall more dense or more sparse, and may exaggerate the amount of activity apparent during the interaction.

User Content

Visualises summary data concerning the contents of user files. The contents themselves are not collected, to protect the user’s privacy, but summary information about how the pattern is used is recorded. This view shows a schematic of the tracker pattern and marks the locations where data (pitch, instrument, volume, panning, and/or effect) is entered. This allows investigation of how the viewport window is affecting musical expression – e.g. whether users confine themselves to smaller areas in the pattern, to avoid having to scroll or hide data. It also indicates how productive a user is, in comparison to the amount of interaction that went into producing the content.



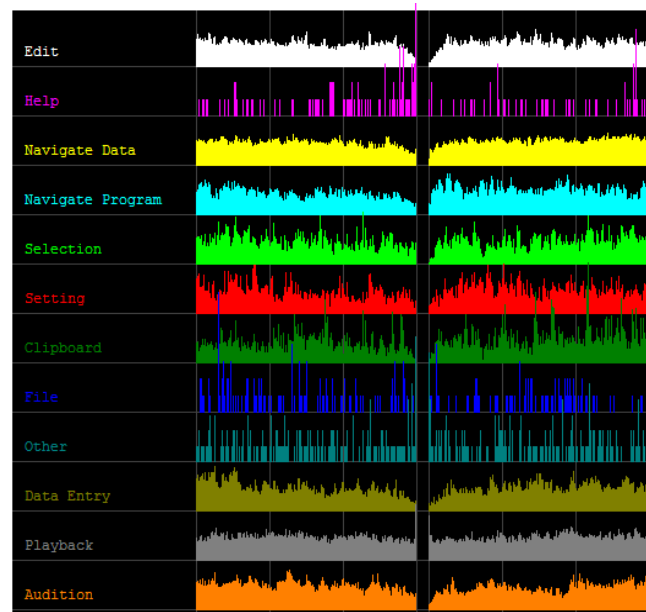
Window Log

```
Window Log
00:00:03.016* window (007603F6h)
00:00:03.016* message (WM_ACTIVATE, wParam=00000001h, lParam=00A702E8h)
00:00:03.016* message (WM_NCACTIVATE, wParam=00000001h, lParam=00A702E8h)
00:00:03.016* message (WM_SETFOCUS, wParam=005D039Ch, lParam=00000000h)
00:00:03.907* window (007603F6h)
00:00:03.907* message (WM_NCACTIVATE, wParam=00000000h, lParam=00A702E8h)
00:00:03.907* window (007603F6h)
00:00:03.907* message (WM_KILLFOCUS, wParam=005D039Ch, lParam=00000000h)
```

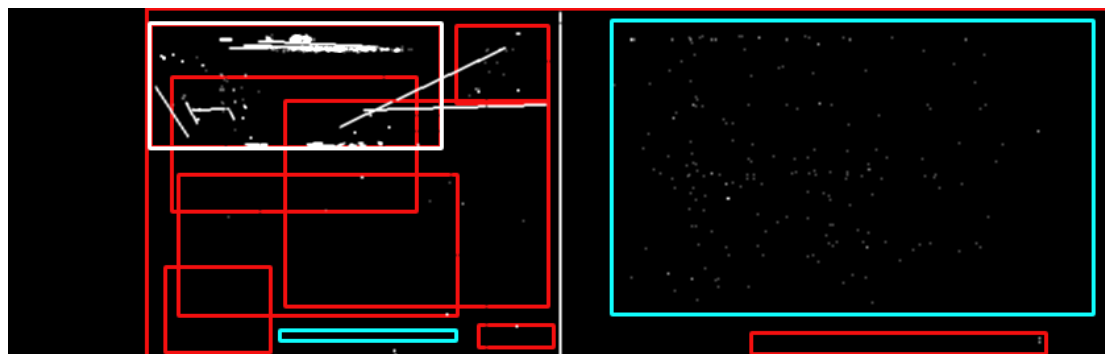
Lists the session log events associated with managing the given window – e.g. windows notifications, such as WM_MOVE and WM_SIZE. Like the map, this view is most helpful in debugging the detection algorithms that workout window statistics, such as the duration of the window.

Distraction Chart

Like the **Context Chart**, the Distraction Chart visualises the contexts of key inputs, but only those surrounding a distraction event (a period of 10s or more inactivity). Instead of contexts, the general types of log entries can also be shown. The events are charted around a central gap, representing the distraction, and use the distraction data generated by the **Distraction Analysis**. The view attempts to highlight events that are more or less associated with (and possibly precipitate) interruptions in interaction.



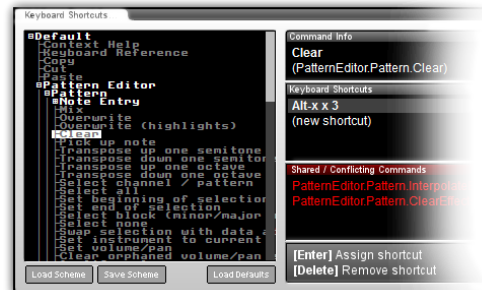
Window Map



This appendix describes updates to the *reViSiT Pro* software, used to maintain and broaden interest in the experiment, subsequent to its launch.

reViSiT 1.1 Pro user-customisable key shortcuts.

Previously, keyboard shortcuts were fixed, to allow the experiment to study how users handle specific key combinations and layouts, enabling easy comparisons between individuals. However, like many aspects of the program, these keys were based on *IT2*, which had the effect of discouraging users from other tracker backgrounds, and also introduced some non-standard keys for common tasks, such as the clipboard. The feature logged changes made to the key assignments, allowing the experiment to look at which default keys are least popular and which changes are most common.

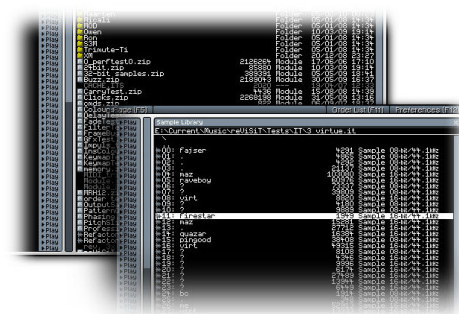
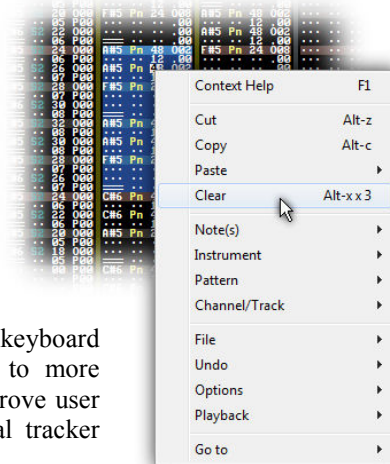


reViSiT 1.2 Pro high-definition, high-resolution pattern editing

High-definition pattern editing addresses a widespread criticism of tracking; that the grid-like notation inherently quantises the music to rigid divisions of time, limiting its musical uses. In truth, trackers offer much finer timing resolution than the displayed grid, but placing events between the rows of the pattern involves the use of the effects column, which makes the process esoteric and visually confusing. This update allows the user to ‘zoom’ into the space between the rows and edit finer-grained music, using standard pattern-editing methods.

reViSiT 1.3 Pro improved mouse support, graphical feedback and direct manipulation

As a tracker, input methods revolve around the keyboard, which can discourage novice users. This update attempted to ease the initial learning curve, adding mouse-based interaction styles with which users may already be familiar (e.g. *drag-n-drop* and direct manipulation techniques – Shneiderman and Plaisant, 2005). This included an ‘info bar’ to provide graphical input and feedback for pattern data (e.g. a piano keyboard for pitch; see Figure 5-1), the ability to select, move and copy blocks of notes using the mouse, and a right-click menu that exposes most of the editor’s functions (annotated with keyboard shortcuts). These methods are designed as a stepping stone to more efficient keyboard interaction methods. The features aim to improve user retention, but also allow the experiment to compare traditional tracker interaction with equivalent direct manipulation methods.



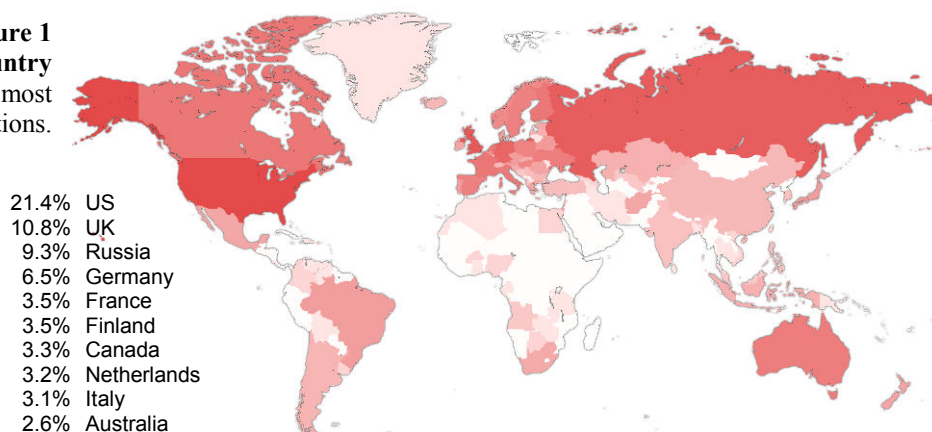
reViSiT 1.4 Pro sample and instrument libraries

While *reViSiT* interaction focuses on the *pattern editor*, where music is edited, other screens exist to handle file loading, arrangement, and program settings. Though these screens are ancillary to the main composition activity, users must use them to load instruments for their song. User feedback and experiment data suggested that standard file dialogs were interrupting workflow. This update integrates file management into the tracker UI, allowing the user to browse instruments or samples on disk, and audition them before loading. It also allows

users to look inside other tracker songs, downloaded from the Internet, and ‘rip’ the samples or instruments for use in their own song. This helps address the fact that *reViSiT* is not supplied with any such media. Such *open interchange* is established in the tracker tradition, and was a key recommendation of the *Creative Support Tools* workshop (Resnick *et al*, 2005).

This appendix provides a brief overview of the user sample and data collected, during the 2 years the experiment was running. To gauge user backgrounds, participants completed a questionnaire during registration (see section 5.3.1). 2,351 surveys were completed, though interaction data from program use was only received from 1,125 users (47.8%), likely due to local firewalls or execution on computers not connected to the Internet.³

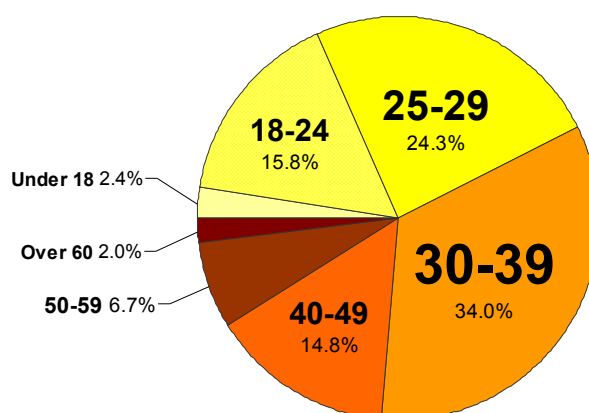
Figure 1
Users by country
inset with 10 most
common locations.



user demographic

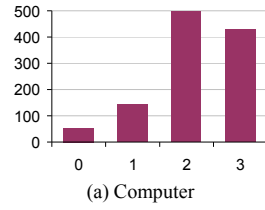
Figure 1 provides the breakdown of participants with respect to location, showing an expected bias towards the English-speaking world, Europe and other technological developed countries with widespread Internet access. A highly pronounced gender gap was also evident, with 97.9% male participants. Age ranges were more balanced, showing the typical bias to young adults seen in technology use, but stronger in over 30s, corresponding to individuals with memories of earlier tracker and demo scenes, and experienced music professionals.

Figure 2
Users by age

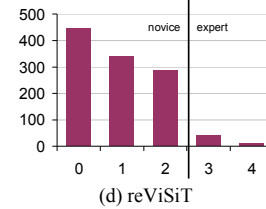
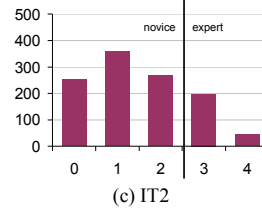
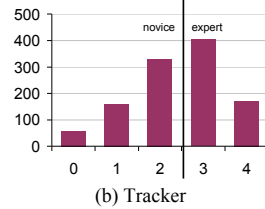


³ This section only summarises the responses of users from which data was received, to illustrate the background of subjects in later analyses. No significant difference, however, was noted between responses (see Figure 4) from users with and without interaction data (based on a *two-tailed Z-test* of mean survey responses, where $\alpha = .05$).

Figure 3
Users by experience



	<i>median</i>	<i>mode</i>	<i>mean</i>
(a) Computer	2	2	2.16
(b) Tracker	3	3	2.42
(c) IT2	1	1	1.49
(d) reViSiT	1	0	0.96



*skills and
experience*

The questionnaire also probed each user's technological background, including prior experience of relevant music technologies that may influence their performance with reViSiT. Four types of experience (summarised in Figure 3) were tested using ordinal scales, where scores of 3 or more define significant experience (expertise):

- **Computer Experience** (48% expert) – assesses a user's comfort with generic computer interaction (keyboard, mouse, software, etc.); indicating widespread, advanced computer skills, beyond that of regular users (1), possibly the result of a higher disposition to tracking among computer literate individuals.
- **Tracker Experience** (51% expert) – gauges prior exposure to tracking concepts or programs (e.g. early trackers, *Fast Tracker*, *Renoise*); indicating widespread skill, with a majority of users stating significant experience.
- **IT2 Experience** (22% expert) – to recognise specific expertise with *Impulse Tracker 2* (see 2.2.1), which may directly benefit reViSiT interaction; showing limited awareness or experience, but 42 experts who still use the DOS program.
- **reViSiT Experience** (5% expert) – to acknowledge prior exposure with earlier versions of reViSiT (e.g. alpha or beta testers); indicating some awareness and playing with the program, but significantly less developed expertise.

Other experience with music and specific music technologies is presented in Figure 4, along with interaction preferences. As a survey of volunteers for the study, responses partly reflect the backgrounds of individuals who were attracted to the reViSiT tracker VST plugin, and interested in alternatives or extensions to their existing setup. As the most common type of VST host, sequencers were familiar to the majority of participants (84.3%), with 53.1% stating expertise. As earlier, experience and expertise of several specific trackers was also identified by participants.

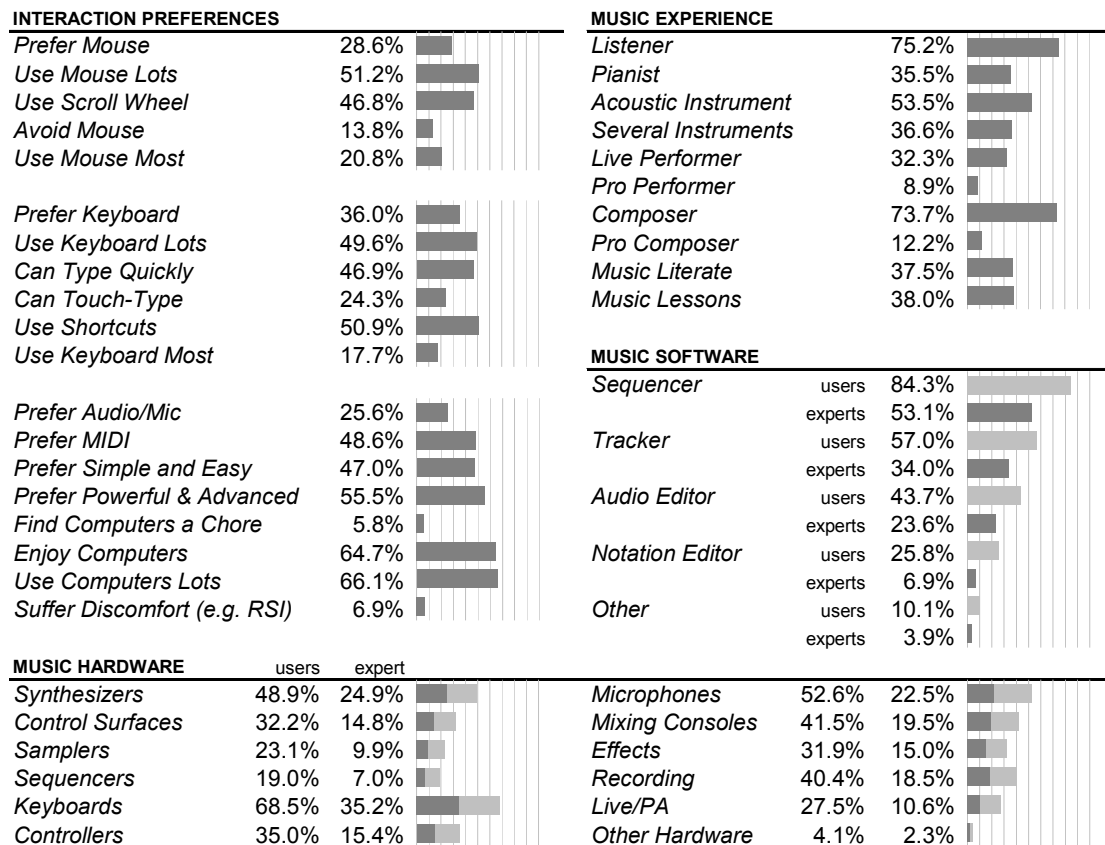


Figure 6 – Interaction preferences and music experience. Percentage of study subjects (n=1125).

preferred
input device

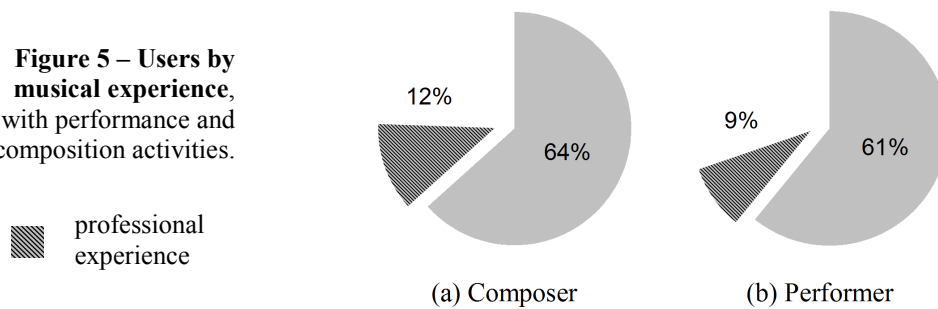
A small, but significant preference for the keyboard, compared to the mouse, was also noted (*two-tailed z-test, $p < .05$*), reflecting the larger role of the keyboard in tracker interaction. Correspondingly, almost half (46.9%) of the participants claimed some typing skill, with 24.3% able to touch-type, without relying on visual feedback. Of those who stated a preference for the keyboard, 44.6% still found themselves making significant use of the mouse. However, both computer input devices were less popular than input through more musically-oriented MIDI devices (48.6%), which allows the user to transfer their musicianship to the digital domain.

music experience

In Figure 5, over two thirds of the sample are shown to have experience with some mode of performing (piano, acoustic, or live: 69%, 9% professionally) or composing (76%, 12% professionally). Over half (50.9%) of all performers have experience of playing the piano, which translates well to MIDI and tracker-based interaction. This matches the widespread use of music keyboards (68.5% of users) in Figure 6.4. Many users also showed familiarity with studio hardware, such as microphones (52.6%), mixers (41.5%), effects (31.9%) and recording equipment (40.4%). However, almost a quarter (23.5%) of all participants had no experience with studio or hardware input

devices, presumably focusing on computer-based editing tools and techniques. Moreover, though most identified themselves as composers, relatively few indicated musical literacy (37.5%) or training (38.0%). This suggests that only around half of the electronic musicians in this study come from formal or traditional music backgrounds, while others simply rely on more informal, self-taught approaches, developing technique simply through use of software or hardware.

Figure 5 – Users by musical experience, with performance and composition activities.



Overall, the 1125 users who took part in the study include a broad sample of digital musicians, including those both with and without performing skills, studio experience, and tracker knowledge, from professional users to desktop hobbyists. Further user traits are discussed in subsequent sections, in context with specific analyses, and with respect to the end-of-experiment questionnaire, detailed in Chapter 9.

interaction data collected

Table 1 summarises the extent of the data collected during the 2 years, 2 months and 11 days in which the experiment was running.

		<i>total</i>								
Users	<i>surveyed</i>	2351	User							
	<i>recorded</i>	1125		<i>mean</i>	<i>median</i>					
	<i>with 30 mins activity</i>	329								
	<i>with 60 mins activity</i>	185								
Sittings	<i>total</i>	5911	5.25	2	Sittings					
	<i>with <1 min duration</i>	1239	1.10	0		<i>mean</i>	<i>median</i>			
	<i>with 30 mins activity</i>	1275	1.13	0						
	<i>with 60 mins activity</i>	678	0.60	0						
Sessions	<i>total</i>	13373	11.89	4	2.26	1	Session			
	<i>with <1 min duration</i>	5077	4.51	1	0.83	1		<i>mean</i>	<i>median</i>	
	<i>with 30 mins activity</i>	1195	1.06	0	0.20	0				
	<i>with 60 mins activity</i>	508	0.45	0	0.09	0				
Duration	<i>total</i>	5912h 33m	5h 15m	24m	1h 5m	15m	26m	3m		
	<i>activity</i>	2376h 30m	2h 7m	14m	24m	6m	11m	2m		
	<i>in reViSiT</i>	1102h 10m	59m	10m	11m	2m	5m	(37s)		
	<i>in Pattern Editor</i>	837h 27m	45m	6m	9m	2m	4m	(24s)		

Table 1 – Summary of user sample. Total users, sittings, sessions and duration recorded during the experiment, plus mean and median average per user and session.

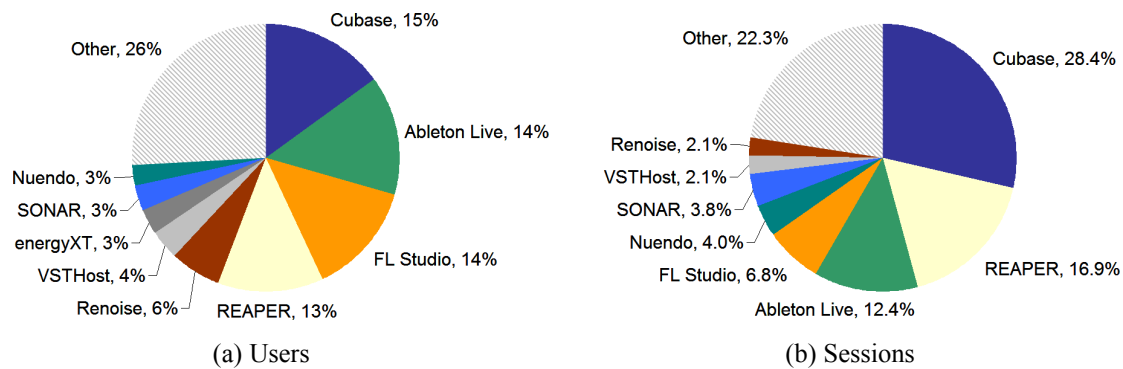


Figure 6 – Hosts used in study. Percentage of (a) users and (b) sessions run in selected hosts.

host compatibility

Though minor bugs were identified in early releases, more users were discouraged by compatibility problems with specific VST plugin hosts, featuring inconsistent or incomplete implementations of the plugin specification – where upon loading the plugin in the host, the user soon discovers an absence of playback, song synchronisation, or keyboard support. Though many of these problems concern freeware, shareware, and open-source music programs, oversights were also noted in commercial and professional software. Figure 6 illustrates the hosts used in the study, dominated by a handful of well-known packages, notably *Steinberg Cubase*, for which *reViSiT* was originally developed.

Such problems are suspected in the 5077 sessions (38% of total) lasting under a minute and the low median session length of 3 minutes. To provide a more accurate picture of a user's exposure to the program, Figure 6.6 also displays figures for individual 'sittings', which concatenate short sessions occurring within minutes of each other, such as when a user loads the plugin several times in quick succession in an attempt to solve configuration problems. With this measure, the median time a user spends in front of *reViSiT* rises to 15 minutes, during which users form an impression of the program.

novice and expert interaction

During the study, the tracker was run for a total of just under 6,000 hours, capturing 2,376 hours of active interaction, in both the host and plugin, ignoring idle periods.⁴ Almost 1,200 sessions longer than 30 minutes of interaction were captured, with half of all users contributing 24 minutes or more. The 185 users who persisted further (with over 60 minutes of interaction) ultimately contributed 90% of the total data collected, and often represented

⁴ Defined as any period longer than 1 minute where no activity or playback is recorded in the log. These periods do not preclude other system activity, beyond the host and *reViSiT* plugin, where data is not collected. Indeed, the interference and distraction provided by other programs (e.g. web browsers, chat clients) potentially constitutes a significant factor in creativity and flow, affecting the user's ability to maintain focus. Though beyond the scope of this research, it is recommended for future study.

individuals with previous tracker or IT2 experience, enabling detailed study of expert interaction and tracker virtuosity. Because of the relatively slow uptake of the program by other musicians, the experiment duration was extended to ensuring sufficient data was captured to study a wider demographic, and provide insight into earlier stages of learning. In total, 391 hours of interaction was captured from tracker novices, with 72 continuing past the hour mark. Moreover, thresholds and normalisation are also used as appropriate, to ensure that analyses are not biased to more prolific or expert users.