

Networked Music Composition and Performance

Slavko Zagorac

Department of Music
Goldsmiths, University of London

PhD Degree Dissertation

3rd May 2024

I declare that the work submitted is my own, and does not contain any unacknowledged material from other sources.

Slavko Zagorac

Abstract

This thesis investigates the potential of computer networks to transform existing music-making relationships within the context of contemporary music composition and performance practice. It consists of two primary components: the development of the ZScore networked music-making environment and a portfolio of compositions written specifically for the ZScore system. The technical capabilities of the system and creative compositional intentions have been evaluated through a series of workshops and performances with musicians and live audiences. This study provides detailed information on the project's technical and creative objectives, their implementation, and research outcomes.

ZScore is a collection of third-party and newly developed components aiming to provide solutions for complex notation authoring, the reliable distribution of interactive score representations to heterogeneous clients over a computer network, precise performance scheduling, and the rendering of dynamic stave-based scores. Furthermore, it aims to facilitate distributed decision-making agency for all participants and provide immersive audience participation through mobile device connectivity. The specifications of the ZScore system have been derived from an analysis of existing networked notation systems and state-of-the-art solutions from other industries where high-throughput, low-latency systems have been successfully deployed.

The accompanying composition portfolio outlines the journey towards a new paradigm of music-making, starting from a traditional static score that is fully defined by a composer and faithfully performed by musicians, and concluding with an interactive networked composition model that provides music-making agency to all participants. The compositional approach deploys an interplay between objective and subjective metamodern methods and explores the entire spectrum between fully composed and freely improvised music. Each composition in the portfolio addresses a set of new challenges, both technical and aesthetic. The composition commentary provides insight into the specific challenges for each score, the strategies employed to address related issues, and the outcomes of the applied solutions.

Acknowledgements

I would like to thank Roger Redgate for his profound knowledge, guidance, insight, and experience, and for being there throughout my PhD journey. I am grateful for the faith he showed in my ideas and research as they developed from project to project. Furthermore, I extend my gratitude to all my secondary supervisors who provided invaluable support during various stages of my research. I am especially grateful to Patricia Alessandrini for her expertise, dedication, and inspiration at the outset of my journey. I then had the privilege of collaborating with Michael Zbyszynski, whose in-depth technical and creative guidelines helped steer my work in the midst of a complex phase of this research. Finally, I would like to thank Jenn Kirby for her invaluable support, advice, and help in my efforts to bring this work to completion, as well as our frequent and most enjoyable discussions on the subjects of music and technology.

I would also like to acknowledge the funding provided by Goldsmiths Music Department and Graduate School. It enabled me to collaborate with a host of exceptional musicians and ensembles, whose invaluable expertise and feedback played a pivotal role in shaping the ZScore system and the overall outcome of this work. I would like to thank Ensemble Interface, Moscow Contemporary Ensemble, the remarkable Ligeti Quartet – particularly its members Patrick Dawkins and Val Welbanks – as well as Heather Roche, Benedict Taylor, Tom Jackson, Anne Han, and Khabat Abas for their exceptional musicianship, patience and open-mindedness in our collaborations.

Publications

Two papers highlighting the features of the ZScore system were presented at the TENOR conference in 2018 and 2020. The first paper focused on the technical aspects of the system, whilst the second paper discussed the dynamic notation overlays implementation.

S. Zagorac and P. Alessandrini (2018). “ZScore: A Distributed System For Integrated Mixed Music Composition and Performance”. In: *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR’18*. Montreal, Canada: Concordia University, pp. 62–70. ISBN: 978-1-5251-0551-7. URL: https://www.tenor-conference.org/proceedings/2018/09_Zagorac_tenor18.pdf

S. Zagorac and M. Zbyszynski (2020). “Networked Comprovisation Strategies with ZScore”. In: *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR’20/21*. Hamburg, Germany: Hamburg University for Music and Theater, pp. 133–140. ISBN: 978-3-00-066930-9. URL: https://www.tenor-conference.org/proceedings/2020/18_Zagorac_tenor20.pdf

Contents

1	Introduction	18
1.1	To Music Is To Communicate	22
1.2	Decision-making Spectrum	24
1.3	Time, Action, And Sound Notation	28
1.4	Music Composition In The Smartphone Era	34
1.5	Networked Music-making	37
1.6	Existing Solutions	44
1.7	Networked Notational Perspective	49
1.8	Gaps And Research Aims	52
1.8.1	Solution Framework	52
1.8.1.1	Relationships between those taking part and the physical setting	53
1.8.1.2	Relationships among those taking part	53
1.8.1.3	Relationships between the sounds that are being made	55
1.8.1.4	Modes of participation	58
1.8.1.5	User Interface Design	59
1.8.1.6	Action Modelling	60
1.8.2	Gaps	63
1.8.3	Objectives	64
2	ZScore	67
3	Portfolio	74
3.1	Red Mass	77
3.1.1	Objective: Compositional Method	77
3.1.2	Approach To Music Composition And Aesthetics	77
3.1.3	Score Commentary	79
3.2	Ukodus	84
3.2.1	Objective: Linear Networked Composition And Performance	84
3.2.1.1	Dynamic Notation	84
3.2.1.2	Alternating Pane Layout	85
3.2.1.3	Time-space Mapping And Synchronisation	87

	3.2.1.4	Notation Authoring	88
	3.2.1.5	Notation Rendering	93
	3.2.1.6	Performance Control	95
	3.2.2	Score Commentary	96
3.3	Vexilla		102
	3.3.1	Objective: Audience Score Visualisation	102
	3.3.1.1	Embedded Scripting	103
	3.3.1.2	Audience Score View	105
	3.3.2	Score Commentary	106
	3.3.2.1	The Use Of Colour In Music Notation	113
3.4	Comprov		114
	3.4.1	Objective: Networked Comprovisation	114
	3.4.1.1	Notation Segmentation	114
	3.4.1.2	Dynamic Notation Overlays	117
	3.4.1.3	Comprovisation Controls	118
	3.4.1.4	Randomisation Strategy	120
	3.4.1.5	Continuous Play	120
3.5	Union Rose		122
	3.5.1	Objective: Interactive Composition And Performance	122
	3.5.1.1	Web ZScore	123
	3.5.1.2	Audience Score Representation	125
	3.5.1.3	Digital Audio	129
	3.5.1.4	Score Scripting Extensions	130
	3.5.1.5	Score State Management	132
	3.5.2	Score Commentary	133
	3.5.2.1	I. AM.	134
	3.5.2.2	I Believe	144
	3.5.2.3	I Want	149
	3.5.2.4	Coda	152
3.6	Socket Dialogues		156
	3.6.1	Socratic Dialogues	156
	3.6.2	Objective: Music-making Role Democratisation	158
	3.6.2.1	Dialogue Roles	158
	3.6.2.2	Generic Instrument Notation	160
	3.6.2.3	Transposition Strategy	163
	3.6.2.4	Score Builder Strategy	164
	3.6.2.5	Audience Feedback Loop	165
	3.6.2.6	Mobile Instrument	167
	3.6.3	Score Commentary	168

3.6.3.1	Pitch	169
3.6.3.2	Rhythm	176
3.6.3.3	Melody	178
3.6.3.4	Timbre	179
3.6.3.5	Improv	180
3.6.3.6	Interlude	182
4	Results And Reflections	186
5	Conclusions	193
	References	195
A	ZScore Server Testing	205
A.1	Testing Scope	205
A.2	Testing Environment	205
A.2.1	Router Setup	206
A.2.2	Wi-Fi Setup	206
A.3	Testing Procedure	208
A.3.1	JMeter Tests	208
A.3.2	Custom Undertow Client Tests	209
A.4	Findings And Analysis	209
A.5	Scalability Ramifications And Possible Solutions	210
A.5.1	System Design For Up To 100 Wi-Fi Users	211
A.5.2	System Design For 200 To ~500 Wi-Fi Users	211
A.5.2.1	Wireless Mesh vs Multiple APs	213
A.5.3	System Design For ~500 To ~1000 Wi-Fi Users	214
A.5.4	System Design For 1000+ Wi-Fi Users	214
A.6	Detailed Test Results	215
B	Participants' Feedback	223
B.1	Musician Feedback Form	223
B.2	Musicians' Feedback	224
B.2.1	<i>Union Rose</i> Workshop 23 Feb 2022	224
B.2.2	<i>Socket Dialogues</i> Workshop 9 Jan 2023	227
B.2.3	<i>Vexilla</i> Musician Feedback 1 May 2018	230
B.3	Audience Feedback Form	233
B.4	Audience's Feedback	233
B.4.1	<i>Union Rose</i> Workshop 23 Feb 2022	233
B.4.2	<i>Socket Dialogues</i> Workshop Audience Feedback 4 Jul 2022	236
B.4.3	<i>Socket Dialogues</i> Workshop, Audience Feedback 9 Jan 2023	238

B.4.4 Analysis Of The Audience Keyword Mention Frequency	241
C Flags Used In <i>Vexilla</i>	242

List of Figures

1.1	Communication with sound and movement in the physical world	23
1.2	Simplified traditional music-making information flow	23
1.3	Music performance decision-making spectrum	25
1.4	Helmut Lachenmann, <i>Pression</i> for one Cellist, score excerpt	29
1.5	Brian Ferneyhough, <i>Time and Motion Study II</i> , score excerpt	30
1.6	Klaus K. Hübler, <i>String Quartet No.3</i> , Violin 2, page 8, score excerpt	32
1.7	Klaus K. Hübler, <i>String Quartet No.3</i> , stave layout	32
1.8	Klaus K. Hübler, <i>String Quartet No.3</i> , tablature notation	32
1.9	Aaron Cassidy, <i>Second String Quartet</i> , Violin 1, score excerpt	33
1.10	Aaron Cassidy, <i>Second String Quartet</i> , stave layout	33
1.11	A networked music-making environment	38
1.12	Multi-directional communication in networked music performance	39
1.13	Network communication models	39
1.14	Network application architectures	41
1.15	Networked notation software summary	45
1.16	Middleware solutions for networked music-making	46
1.17	Relationships between participants and physical setting	53
1.18	Western classical music participant relationships	54
1.19	Networked score relationships	54
1.20	Networked music-making participant relationships	55
1.21	Composition process for networked scores	55
1.22	Networked scores composition material	56
1.23	Composition material generation process	56
1.24	Composition time structure components	57
1.25	Composition vertical structure	57
1.26	Music material dynamic assignment	58
1.27	Modes of participation in a networked performance	58
1.28	Intuitive UI design principles	59
1.29	Sequence diagram for instant action outcome	61
1.30	Sequence diagram for delayed action outcome	62
1.31	Compositional intentions, UI design, and action model relationships	62

2.1	ZScore performance system components	68
2.2	ZScore high-level software component architecture	69
2.3	Disruptor processing structure	69
2.4	Disruptor data access sequencing	70
3.1	<i>Pierrot Lunaire</i> poems (3 x 7)	79
3.2	<i>Pierrot Lunaire</i> rondel poem structure	80
3.3	The structure of <i>Red Mass</i>	80
3.4	<i>Red Mass</i> section info	81
3.5	<i>Rote Messe</i> , first beat pitches as the chord and normalised pitch set	81
3.6	<i>Red Mass</i> , opening ascending sequence	82
3.7	Cello part opening parallel fifths in <i>Rote Messe</i>	82
3.8	<i>Red Mass</i> , material “A” piano chord and derived intervals	82
3.9	<i>Red Mass</i> , Material “B” pitch set	83
3.10	Alternating pane layout for a multiple stave full score and a single stave part	86
3.11	Alternating pane time windows	87
3.12	<i>Ukodus</i> , flute part excerpt illustrating Beat Lines	87
3.13	ZScore hierarchical score layer entity relationships	88
3.14	Bar layer content	88
3.15	Adobe Illustrator layer structure	89
3.16	Notation symbol library for Adobe Illustrator	90
3.17	ZScore Tools plugin for Adobe Illustrator	90
3.18	<i>Ukodus</i> , an example of the score page containing mixed notation	91
3.19	<i>Ukodus</i> , flute stave example	91
3.20	<i>Ukodus</i> , clarinet stave example	92
3.21	<i>Ukodus</i> , cello stave example	92
3.22	<i>Ukodus</i> , piano stave example	93
3.23	<i>Ukodus</i> , cello part dynamic notation in INScore with alternating pane layout	93
3.24	<i>Ukodus</i> , full score in alternating pane layout	94
3.25	<i>Ukodus</i> , ZScore Performance Control GUI	95
3.26	<i>Ukodus</i> , Sudoku puzzle used for the composition material	97
3.27	<i>Ukodus</i> , Sudoku puzzle W → E upper contour	97
3.28	<i>Ukodus</i> , Sudoku puzzle W → E upper values	98
3.29	<i>Ukodus</i> , Sudoku puzzle W → E lower values	98
3.30	<i>Ukodus</i> , basic structure	98
3.31	<i>Ukodus</i> , section 1 structure	99
3.32	<i>Ukodus</i> , sections 2 and 3 structure	99
3.33	<i>Ukodus</i> , Group 1 events in section 1 part 1	100
3.34	<i>Ukodus</i> , Group 2 events in section 1 part 1	100

3.35	Roman Vexillum	102
3.36	<i>Vexilla</i> staging with synchronised animations projected onto a canvas	103
3.37	<i>Vexilla</i> , AV part	103
3.38	Adobe Illustrator layer structure for embedded scripts	104
3.39	<i>Vexilla</i> , Structure	107
3.40	<i>Vexilla</i> , flag feature mapping to sound	108
3.41	<i>Vexilla</i> , <i>Na Planincah</i> , traditional folk song from Slovenia	108
3.42	<i>Vexilla</i> , part 1 Structure	108
3.43	<i>Vexilla</i> , Cello part using <i>Na Planincah</i> melody elements	109
3.44	<i>Vexilla</i> , part 1 star gesture	110
3.45	<i>Vexilla</i> , section 1 parts structure	111
3.46	<i>Vexilla</i> , sections 2 and 3 parts structure	112
3.47	<i>Vexilla</i> , use of colour in the notation	113
3.48	<i>Comprov</i> , string instrument notation layout	115
3.49	<i>Comprov</i> , vertical stave layout	116
3.50	<i>Comprov</i> , Cello notation excerpt	117
3.51	<i>Comprov</i> , dynamic notation overlays for position, speed, pressure, and dynamics	118
3.52	<i>Comprov</i> , dynamic pitch notation overlay	118
3.53	<i>Comprov</i> , Control GUI	119
3.54	Randomisation strategy configuration control	120
3.55	Union Chapel Rose	122
3.56	Web ZScore JavaScript libraries	124
3.57	<i>Union Rose</i> , cello part in Web ZScore	124
3.58	ZScore JavaScript audio libraries	126
3.59	<i>Union Rose</i> , stylised SVG rose for the audience view	126
3.60	<i>Union Rose</i> , one of the stylised SVG shapes	127
3.61	<i>Union Rose</i> , shape groups	127
3.62	<i>Union Rose</i> , SVG rose cover grid consisting of 64 tiles	128
3.63	ZScore Max mxj and jsui objects	129
3.64	ZScore Max patch	130
3.65	<i>Union Rose</i> , new scripting resource types	131
3.66	<i>Union Rose</i> , high level structure	133
3.67	<i>Union Rose</i> , tempo curve	134
3.68	Ligeti, <i>2nd String Quartet</i> , bars 19-22 harmonic structure	135
3.69	Harmonic structures extracted from Ligeti's <i>2nd String Quartet</i>	135
3.70	<i>Union Rose</i> , section tetrachords	136
3.71	<i>Union Rose</i> , pitch material, first four bars	136
3.72	<i>Union Rose</i> , pitch material, bars 5 - 8	137
3.73	<i>Union Rose</i> , Page 2 score excerpt	137

3.74	<i>Union Rose</i> , audience “welcome” screen	138
3.75	<i>Union Rose</i> , audience view, innermost tile circle	139
3.76	<i>Union Rose</i> , audience view, “centreShape” assembly	139
3.77	<i>Union Rose</i> , section <i>I. AM. A'</i> audience view	140
3.78	<i>Union Rose, I. AM. A'</i> section instrumentation	142
3.79	Max MC Groove harmonic series preset menu	143
3.80	Coltrane’s “A Love Supreme” four-note chant	144
3.81	<i>Union Rose, I Believe</i> section pitch material	144
3.82	<i>Union Rose</i> , Violin 2, section B first bar	144
3.83	<i>Union Rose</i> , Violin 2, section B second bar	145
3.84	<i>Union Rose, I Believe</i> , section B’ audience view	146
3.85	<i>Union Rose, I Believe</i> , section B’ speech texture components	148
3.86	“Money, Money, Money” chorus melody	149
3.87	<i>Union Rose</i> , Viola part, section C notation	149
3.88	<i>Union Rose</i> , Cello part, section C, last page	150
3.89	<i>Union Rose, I Want</i> , section C’, audience view	150
3.90	<i>Union Rose, I Want</i> , section C’ instrument-page randomisation table	151
3.91	<i>Union Rose, I Want</i> , section C’ digital audio texture	152
3.92	<i>Union Rose, Coda</i> tempo curve	153
3.93	<i>Union Rose, Coda</i> audience view, selected tiles animation	154
3.94	<i>Union Rose, Coda</i> , audience view end screen	155
3.95	Plato’s dialogues scale	157
3.96	Aspects of music mapped to the circle of fourths	157
3.97	<i>Socket Dialogues</i> , web score, instrument transposition selection	159
3.98	<i>Socket Dialogues</i> , Presenter selection	159
3.99	<i>Socket Dialogues</i> , Presenter selection in progress	160
3.100	<i>Socket Dialogues</i> , Control GUI order of play	160
3.101	<i>Socket Dialogues</i> , web score role choice	161
3.102	<i>Socket Dialogues</i> , stave layout	161
3.103	<i>Socket Dialogues</i> , timbre notation	162
3.104	<i>Socket Dialogues</i> , pitch line notation rules	162
3.105	<i>Socket Dialogues</i> , named pitch line notation example in concert pitch	163
3.106	<i>Socket Dialogues</i> , named pitch notation transposed to B ^b	163
3.107	Enhanced Adobe Illustrator score export plugin	164
3.108	Named pitch implementation in Adobe Illustrator	164
3.109	<i>Socket Dialogues</i> , voting audience view	166
3.110	<i>Socket Dialogues</i> , Web Score audience vote view	166
3.111	<i>Socket Dialogues</i> , Control GUI audience vote view	167
3.112	<i>Socket Dialogues</i> , voting audience view	168

3.113	<i>Socket Dialogues, Pitch</i> dialogue structure	169
3.114	<i>Socket Dialogues, Pitch</i> , opening statement	170
3.115	<i>Socket Dialogues, Pitch</i> , interval “harmoniousness”	170
3.116	<i>Socket Dialogues</i> , opening page for all musicians who are not the Presenter	170
3.117	<i>Socket Dialogues, Pitch</i> , Dissent part reaction to the Presenter’s statement	171
3.118	<i>Socket Dialogues, Pitch</i> , bar 10 notation	171
3.119	<i>Socket Dialogues, Pitch</i> , Dissent, bar 12 notation	172
3.120	<i>Socket Dialogues, Pitch</i> , Audience audio score, tonal centre	172
3.121	<i>Socket Dialogues, Pitch</i> , Audience audio score, perfect fourth	173
3.122	<i>Socket Dialogues, Pitch</i> , Audience audio score, perfect fifth	173
3.123	<i>Socket Dialogues</i> , Control GUI, audience view controls	176
3.124	<i>Socket Dialogues, Rhythm</i> block durations in number of beats	177
3.125	<i>Socket Dialogues, Rhythm</i> , Present, bar 1 notation	177
3.126	<i>Socket Dialogues, Rhythm</i> , Present, bar 5 notation	177
3.127	<i>Socket Dialogues, Melody</i> , Presenter’s theme	178
3.128	<i>Socket Dialogues, Melody</i> dialogue structure	179
3.129	<i>Socket Dialogues, Melody</i> , Presenter’s theme inversion	179
3.130	<i>Socket Dialogues, Melody</i> , Presenter’s theme inversion in Web Score notation	179
3.131	<i>Socket Dialogues, Timbre</i> dialogue structure	179
3.132	<i>Socket Dialogues, Timbre</i> , Presenter’s opening theme	180
3.133	<i>Socket Dialogues, Timbre</i> , Presenter’s bar 10 score	180
3.134	<i>Socket Dialogues, Improv</i> dialogue structure	181
3.135	<i>Socket Dialogues, Improv</i> , Presenter’s theme free improvisation	181
3.136	<i>Socket Dialogues, Improv</i> , tempo curve	181
3.137	<i>Socket Dialogues</i> , Control GUI presets	182
3.138	<i>Socket Dialogues, Interlude</i> , audience view	183
3.139	<i>Socket Dialogues</i> , Control GUI, audience instrument configuration	183
3.140	<i>Socket Dialogues</i> , audience instrument preset values	183
3.141	<i>Socket Dialogues, Interlude</i> web score	184
3.142	<i>Socket Dialogues, Interlude</i> web score overlays	184
3.143	<i>Socket Dialogues</i> , text message controls	185
3.144	<i>Socket Dialogues</i> , text message in Web Score	185
A.1	Test Environment	206
A.2	Router bridge ports assignment	207
A.3	Guest bridge firewall rules	207
A.4	Guest bridge NAT rules	207
A.5	Router Queue setup	207
A.6	System architecture for up to 100 Wi-Fi users	211

A.7 System design for up to 200 Wi-Fi users 212
A.8 System design for 500 Wi-Fi users 213
A.9 System design for 1000+ Wi-Fi users 214

B.1 Musicians' Feedback Form 223
B.2 Audience Feedback Form 233
B.3 Audience feedback keyword analysis 241

List of Definitions

Client a computer program or device that requests and receives services from another program or device called a server over a network.

CPN (Common Practice Notation) a standard form of music notation used across Western musical traditions.

CSS (Cascading Style Sheets) a language specifically designed to style the presentation of a document written in markup language like HTML.

Ethernet a family of wired computer networking technologies used to connect devices over a network.

GSAP (GreenSock Animation Platform) a JavaScript animation library developed by GreenSock.

HTML (HyperText Markup Language) a standard markup language used for creating Web pages.

HTTP (Hypertext Transfer Protocol) an application layer protocol within the Internet protocol suite for transmitting hypermedia documents like HTML.

iOS an operating system used for mobile devices manufactured by Apple Inc.

IP (Internet Protocol) a core protocol for addressing and routing data packets across networks, ensuring that they reach their intended destination.

Java a platform-independent, general-purpose programming language.

JavaFx a software platform specifically designed for creating and delivering rich client applications written in Java.

JavaScript an interpreted programming language – one of the core technologies of the World Wide Web.

JSON (JavaScript Object Notation) a human-readable file format for exchange of data between applications.

LAN (Local Area Network) a computer network connecting devices within a limited physical space, such as a home, office, or concert venue.

Max a visual programming environment for creating music and multimedia content (Puckette and Zicarelli 1990).

Max external a custom-coded object that extends the functionality of Max software beyond its built-in features.

Max patch the fundamental unit of creation in a Max visual programming environment.

MOM (Message Oriented Middleware) a software or hardware infrastructure that facilitates the asynchronous exchange of data between networked applications.

NTP (Network Time Protocol) a network protocol specifically designed to synchronise the clocks of networked devices.

OSC (Open Sound Control) a networking protocol enabling control of audio equipment, computers, and other multimedia devices in various applications, including music performance.

P2P (Peer-to-peer) a distributed application architecture where devices or programs communicate directly with each other without relying on a central server.

Raster graphics a digital image format composed of a grid of small squares called pixels.

Server a computer program or device that makes resources, data, or services available to other computers or programs, known as clients, over a network.

SPL (Sound pressure level) a logarithmic measure of the effective pressure of a sound relative to a reference value, defined in dB (decibel).

SSE (Server-Sent Events) a server push technology that enables a web client to receive automatic updates from a server via an HTTP connection.

SVG (Scalable Vector Graphics) a human-readable, vector image format for defining two-dimensional graphics.

TCP (Transmission Control Protocol) a reliable communication protocol that provides data delivery in the correct order between applications running on networked devices.

Tween a fundamental building block of animation within the GreenSock Animation Platform (GSAP) library.

UDP (User Datagram Protocol) a connectionless protocol prioritising speed over guaranteed delivery for data transfer over networks.

UML (Unified Modeling Language) a standardised modelling language for visualising system design.

VLAN (Virtual Local Area Network) a logical grouping of devices on a physical network that allows them to communicate as if they were on a separate network segment.

WebSocket a communication protocol that enables a two-way real-time connection between a web client (such as a web browser) and a web server.

Wi-Fi a wireless networking technology that uses radio waves to connect electronic devices to a network.

Chapter 1

Introduction

The implementation of computer networks in music-making practice has the capacity to evolve existing processes of music composition and performance. However, networked performances often encounter technical issues that hinder the full realisation of musical intentions. Inspired by successful deployments of high-throughput, low-latency messaging systems in other industries, I set out to develop a robust, scalable music-making system that delivers consistent performance over local area networks, regardless of the performance venue size or the number of musicians involved.

Further exploration of the initial idea led to a more fundamental inquiry into the potential of networking technology to transform music-making relationships, as defined by Christopher Small (1998) and further explained in Chapter 1.8.

Consequently, this research seeks to answer following questions:

1. How can the use of real-time event-driven networked systems impact music-making relationships and evolve existing processes of music composition and performance?
2. Can innovations in dynamic notation, score structuring, and communications within a networked environment lead to authentic musical aesthetics?

The research resulted in innovations in several areas, as listed below:

1. Multi-directional communications (1.5, 3.5, and 3.6):
 - Real-time collaboration: All participants in a performance, including the audience, can send and receive data and events over a network in real-time.
 - Flexible event targeting: Events can be sent to individual participants, groups, specific participant types (e.g., musicians, audience, conductor), or all participants simultaneously.
2. Participation agency (1.8, 3.5, and 3.6):
 - Distributed decision-making: All participant types (composer, conductor, musicians, audience) can be assigned decision-making agency according to the score's implementation.
 - Role democratisation: Individual participants can take on different roles. For exam-

ple, audience members can assume a sound production agency or collectively decide what notation should be played next; musicians can change the instrumentation or choose which part they wish to play; the conductor can initiate or modify digital sound output, effectively becoming a performer.

3. Dynamic score model (1.8 and 3.2 to 3.6):

- **Extended score:** A ZScore score is a collection of data and algorithms. Score data can include music notation, audio, graphic, and configuration files, whilst server and client algorithms define dynamic score representations and interactive behaviours.
- **Multiple score representations:** ZScore provides proprietary score representation for each participant type (e.g., conductor, musicians, audience, and digital engines), whilst individual score views can be generated within a participant type (e.g., each musician can view their part, and each audience member can be presented with a different score view based on their actions).
- **Score-specific dynamic outcomes:** The server processes all incoming events based on score-specific logic. This may result in updates to the score state, audio-visual output, or targeted modifications to individual participants' score representations.

4. Notation:

- **Alternating pane layout:** A notation layout familiar to classically trained musicians with current position tracking and clear update time window definitions that enable sufficient preparation time (3.2.1.2).
- **Dynamic overlays:** The notation overlay implementation allows for real-time control of the improvisation spectrum between static and dynamic compositional decision-making (3.4.1.2 and 1.2).
- **Integrated composition:** Unconstrained music notation (symbolic and graphic) for mixed ensembles (acoustic and digital instruments) with integrated scripting that manages dynamic score views and interactive behaviours (3.2 to 3.6).

5. System design and architecture (further explained in Chapter 2):

- **Modular server design:** The ZScore server can be split into multiple processes, facilitating horizontal system scalability.
- **Ordered, lock-free event processing:** The server processes events from heterogeneous clients in the order of their arrival, whilst maintaining lock-free, single threaded execution of the composition logic for optimal performance.
- **Network segmentation:** The local network is divided into internal and external VLANs, enhancing security in public performances.
- **Server-side score state management:** This design ensures a consistent view of the score data across all connected clients at any given time.
- **Distributed score processing:** The score processing logic is divided between the server and various clients, optimising performance and reducing network data transmission.

- Multiple Transport instances: ZScore’s Transport object enables tempo-specific event scheduling. Multiple Transport instances can facilitate polytemporal music-making.
6. Creative (3):
- Concept-driven composition: A compositional method that utilises a structured mapping process between predefined concepts and the composition’s structure and material (3.1.2).
 - Subjective-objective interplay: A metamodern approach that utilises emotional and subconscious subjective responses to predefined formal concepts and compositional structures (3.1.2).
 - Dynamic score modelling: A creative process that generates diverse compositional material (e.g., music notation, digital audio, animated visuals, gestures, and scripts), defines interactive behaviours, and models participants’ actions and their outcomes within the composition context (3.2 to 3.6).
 - Composer-performer role: The development of a role where a composer/conductor can actively participate in and shape a performance in a decision-making or sound production capacity (3.6.3.6).

Custom software applications developed specifically for this research project include:

1. ZScore server (2)
2. ZScore Control GUI (3.2.1.6, 3.4.1.3, and 3.6.3.6)
3. Client-side libraries:
 - Adobe Illustrator JavaScript plugin (3.2.1.4)
 - Max mxj external, jsui, and various patches (3.5.1.3)
 - INScore JavaScript libraries (3.2.1.5)
 - Web browser JavaScript libraries (3.5.1.1 and 3.5.1.2)

The system also relies on third-party software, such as Adobe Illustrator (score authoring), INScore standalone client (score visualisation), web browsers (score visualisations), and Max (digital audio processing).

This introductory chapter outlines the context for my research project. Computer networks facilitate communication between all connected nodes and enable distributed, event-driven logic execution. To establish the potential impact of computer networks on music-making practice, I first investigate two key aspects: modes of communication (1.1) and the nature of decision-making (1.2) in music composition and performance processes.

In Chapter 1.3, I examine the work of several composers who have influenced my practice, particularly their innovations in music notation. The notation style employed in my networked portfolio scores draws upon the compositions discussed in this chapter.

To situate my composition practice within contemporary culture, Chapter 1.4 examines the use of digital technology in participatory music-making and explores different types of

digital scores. The development of the portfolio compositions is significantly influenced by two concepts outlined in this chapter: the utilisation of a digital score as a communication interface and the aim of achieving meaningful performative involvement for all participants.

Chapter 1.5 delves into the technical aspects of computer networks relevant to music-making. It explores various communication and synchronisation protocols, their functionality, and potential challenges in this context. Furthermore, the chapter presents various system architectures suitable for networked performances, informing the development of both the current and future iterations of the ZScore system.

In Chapter 1.6, I provide an overview of existing networked music notation systems and related software, focusing on the most prominent and widely used implementations. Informed by this analysis, ZScore's design builds upon existing solutions by addressing identified issues and incorporating select features from these systems, including entire solutions for specific tasks. For example, the INScore standalone client initially handled dynamic notation rendering before the development of ZScore's proprietary web client.

In networked performance systems, a score can become more than static music notation. In Chapter 1.7, I outline the concept of networked notational perspective, which forms the theoretical foundation for the development of the interactive scores in the accompanying portfolio.

The concluding chapter of the introduction (1.8), presents the methodology I employed to address the research questions and provides a detailed breakdown of both the technical and creative objectives. It starts by engaging with three questions posed by Christopher Small (1998, p. 193), which interrogate the fundamental nature of music-making relationships. These questions are subsequently analysed and explored within the context of networked music composition and performance. Additionally, this chapter outlines the principles for the user interface and action model design employed in the portfolio pieces. The specific details of each portfolio composition are presented in Chapter 3.

1.1 To Music Is To Communicate

This chapter investigates modes of communication in music-making, grounded in the observation that music can be understood as an activity in which people take part. According to Christopher Small, using the verb “*to music*” or its gerund form “*musicking*” is a more appropriate way to describe the essence of what happens during a musical performance (Small 1998). The meaning of music cannot be derived exclusively from the musical material or the “work” itself; it requires an expanded horizon that includes all the relationships between different actors in the music-making process:

“The act of musicking establishes in the place where it is happening a set of relationships, and it is in those relationships that the meaning of the act lies. They are to be found not only between those organised sounds which are conventionally thought of as being the stuff of musical meaning but also between the people who are taking part, in whatever capacity, in the performance; and they model, or stand as metaphor for, ideal relationships as the participants in the performance imagine them to be: relationships between person and person, between individual and society, between humanity and the natural world and even perhaps the supernatural world.” (Small 1998, p. 13)

In order to create relationships between music-making participants it is necessary to establish communication channels that convey the information required for the enactment of specific participant roles. The information content that is consumed and produced by each participant role differs. For example, a composer produces instructions for musicians encoded as a symbolic notation, a conductor sends physical gestures to musicians based on a composer’s score, musicians consume the notation and conductor’s gestures and produce sound waves that are heard not only by the audience but also by all other performance participants, who react to generated sound in real-time. Therefore, Small’s observations about the essence of music-making can be expanded by stating that in order “*to music*”, we have “*to communicate*”.

In its essence, computer networking technology facilitates communication between all connected participants and is thus an effective tool for enabling extended modes of communication in music-making practice. This thesis examines existing communication flows in music-making and proposes solutions for extended modes of communication facilitated by computer networks.

Our modern communication systems are the result of humanity’s ancient relationship with sound and movement (Figure 1.1). The ultimate aim of music-making is to create an aural experience for the audience and other performance participants (Kraus and Slater 2016). The pressure of sound waves detected by the listener’s ear sensors triggers complex processing in multiple areas of the human brain, leading to physical reactions and subconscious emotions, as well as conscious feelings and thoughts (Weinberger 2004). Music-making activity is built

¹Source: Kraus and Slater 2016

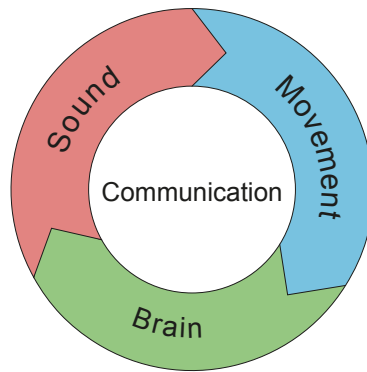


Figure 1.1: Communication with sound and movement in the physical world¹

upon these deeply ingrained communication processes and, in a way, serves as a direct abstract communication link between human consciousnesses. An experienced musician can evoke a mental representation of a sound in isolation, without any external communication, simply by looking at a musical score. However, this is akin to recalling memories stored in the brain from previous lived experiences and is not identical to the actual experience itself (Small 1998, p. 113).



Figure 1.2: Simplified traditional music-making information flow

In Western classical music tradition, the communication is typically conceived of as a one-directional flow (Figure 1.2). In this case, a composer creates a music score in isolation and communicates it to musicians in the form of a printed document. Within a performance setting, musicians generate sound based on the information contained in the score. Performers may also produce conscious or subconscious visual gestures that are not explicitly outlined in the score but still play a significant role in the performance context. The sound waves and visual information produced during the performance are consumed by all participants, including audience members, leading to complex cognitive and physical reactions. It's important to note that this simplified communication scenario lacks a direct feedback loop, apart from the applause at the very end of the performance, serving as an indicator of audience appreciation. This process may involve more complex communication flows if the composer and performers happen to live in the same time period. Prior to the 20th century, any such communications would have been asynchronous to a performance time.

All participants in a music performance constantly evaluate the received aural and visual signals and react based on their respective roles and the performance context. Like any form of communication, music-making requires a certain degree of shared knowledge and experience between the communicators in order to be understood and evaluated adequately.

Computer networks, with their ability to provide real-time, multi-directional communication between all connected participants, are ideally suited as tools for extending existing music-making communication modes.

1.2 Decision-making Spectrum

In addition to the modes of communication explored in the preceding chapter, computer networks may also be applied to transform decision-making processes. This chapter investigates types of decisions and their temporal aspects within the context of music composition and performance.

Composition and improvisation are often regarded as mutually exclusive music-making categories. In practice, however, it is not possible to define a clear boundary between composed and improvised music. Improvisation has been a ubiquitous feature of music-making throughout history. Nevertheless, due to the increasing size of orchestras and the complexity of the works performed, it was largely excluded from Western classical music tradition in the latter half of the 19th century. The influence of jazz and the emergence of avant-garde composers in the second half of the 20th century, such as Christian Wolff, La Monte Young, and Cornelius Cardew, reintroduced improvisational techniques in both contemporary music composition and performance.

Bhagwati (2013a, p. 99) argues that no score can totally determine all aspects of a musical performance and that some elements of music-making will always be contingent. Likewise, a performer's free improvisation is built on years of practice and performance, stemming from a particular tradition and aesthetic context. A free improviser, whether consciously or subconsciously, adheres to a set of rules and regulations specific to the genre. Consequently, any music performance lies somewhere on the spectrum between fully composed (predetermined) and purely improvised (indeterminate). The portmanteau word "comprovisation" is often used to describe this mix.

A piece of music, whether composed or improvised, is conceived through a decision-making process defining what sound or action is to be performed and at what time. In a composed piece, most music material decisions are made pre-performance in isolation by a composer who preserves these decisions as notation realised in a static score. Even in the most meticulously notated scores, however, many performance decisions are left to the performers who interpret the given notation based on their experience and knowledge of the particular music tradition.

Composers may choose to introduce indeterminacy in both the composition and performance processes. John Cage defined indeterminacy as "the ability of a piece to be performed in substantially different ways" (Pritchett 1996). Music compositions, like John Cage's *Music of Changes* (1951), can be constructed through the utilisation of random procedures. This type of compositional indeterminacy still results in a fixed score, which is fully determined prior to the performance. The indeterminacy in a performance can be introduced through various open forms as well as the choice of notation as discussed further in this chapter.

Rodrigo Constanzo (2019) developed a formal methodology for the analysis of the decision-making process in improvised works. His segmentation of music-making decisions into material, formal, interface, and interaction illustrates the improviser's real-time dynamic decision-making approach in an interactive group performance environment.

My work focuses on music-making that exists within the improvisation decision-making spectrum between predetermined statically notated decisions and dynamically made decisions in real-time (Figure 1.3). ZScore is an attempt to provide a platform for unconstrained positioning of the decision-making dial displayed in Figure 1.3, as required by the context of a performed composition. The analysis of the improvisation spectrum required observations of who the decision-makers are, the type of decisions made, and their impact on the aesthetic evaluation of music.

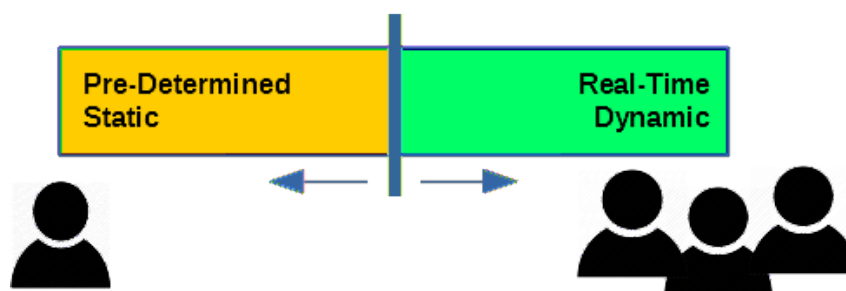


Figure 1.3: Music performance decision-making spectrum

The decision-making spectrum can be translated into the measurement of degrees of freedom assigned to performers in the score implementation. Composers can intentionally imply interpretation freedom through the choice of a notation type (e.g. graphic notation) or provide structural flexibility through open forms that allow for the reordering of music events in time (non-linear composition).

Another consideration related to decision-making in indeterminate scores is a distinction between the actions of choice and chance. A choice implies the performer's active participation in decision-making, whilst chance utilises an external randomisation process, such as a roll of the dice, to determine the sequence of events or define the composition material itself.

Musical Dice Games (Musikalisches Würfelspiel), popular in the 18th century, were an early example of indeterminate scores. The order of play of the pre-composed options in these games was decided by a throw of the dice. A number of composers, famously including Mozart (Hedges 1978), wrote music material for Musical Dice Games. In this context, structural decision-making is always done ahead of a performance, with the responsibility distributed between the composer and performers (dice rollers).

A group of American composers (Cage, Feldman, Wolff and Brown), embraced concepts of choice and chance in the early 1950s (Nyman 1999). Cage extensively used chance as a compositional device, whilst other members of the group used choice to evolve the nature of a music score. Inspired by the mobile sculptures of Alexander Calder and painting techniques

of Jackson Pollock, Earle Brown strived to introduce more spontaneity in performance and mobility in scoring techniques (Brown 2008). This resulted in the introduction of a “mobile” composition concept with a dual nature, described by Brown as:

- a ‘mobile’ score subject to the physical manipulation of its components, resulting in an unknown number of different, integral, and ‘valid’ realisations.
- a *conceptually* ‘mobile’ approach to basically fixed graphic elements; subject to an infinite number of performance realisations through the involvement of the performer’s immediate responses to the intentionally ambiguous graphic stimuli relative to the conditions of performance involvement. (Brown 1954)

Brown also observed that notation employing discrete values could not adequately represent the continual change of sound’s physical attributes (e.g. frequency, amplitude, etc.) in time. This limitation of Western classical symbolic notation and the desire to release the performer’s creative potential culminated in the seminal graphic notation works *December 1952* (1952) and *4 Systems* (1954) radically shifting the dial towards the musicians’ real-time decision-making.

Witold Lutosławski’s concept of a “mobile” differs somewhat from the definitions outlined by Brown. Lutosławski was interested in removing the temporal constraints imposed by a singular score tempo by distributing the decision-making agency for event timing to individual players, whilst still retaining some control over the vertical harmonic structures within the temporal framework. For his *String Quartet* (1964), Lutosławski intentionally created only four instrument parts and not the full score. When musicians insisted on having the full score in order to prepare for live performances, Lutosławski’s wife allegedly cut up individual parts and placed them in boxes which Lutosławski called “mobiles” (Rae 1999). Lutosławski used the indeterminate poly-temporal approach in several orchestral works, such as *Jeux Vénitiens* (1961) and *Symphony No. 2* (1965 - 67). In these works, all parts were fully written out, thereby preserving the harmonic integrity. The individual musician’s freedom of interpretation was limited to the timing and phrasing of the notated parts.

The indeterminate music composition implementations are frequently referred to as “open forms”, an ambiguous term overloaded with multiple meanings. The term was first used by Swiss art critic Heinrich Wölfflin in his attempt to formalise the difference between Baroque and Renaissance visual art. Wölfflin describes close form as “a self-contained entity, pointing everywhere back to itself”, whilst open form “points out beyond itself and purposely looks limitless, although, of course, secret limits continue to exist, and make it possible for the picture to be self-contained in the aesthetic sense” (Wölfflin et al. 1915, p. 204).

Some music theorists still use the term “open form” when discussing scores written before 1950 that do not adhere to strict forms defined by classical music theory, such as a fantasy (Borio and Carone 2018). When referring to post-1950 music, Blumröder uses the term to describe (1) form that has no inevitable beginning or end, or (2) form with variable formal shape (Blumröder 1984). This line of thinking draws from Stockhausen’s writing on “Momentform” (Stockhausen

1963) which differentiates between an open (polyvalent) form offering a choice regarding the order of the score elements in time, such as in *Klavierstück XI* (1956), and a score which does not have a predetermined start, end, or duration, such as *Zyklus* (1959).

Klavierstück XI for piano consists of 19 notated fragments laid out on a single large page. The performer may choose to begin with any fragment and freely continue to any other until one of the fragments has been played three times. Additionally, markings for tempo, dynamics, and articulation at the end of each fragment indicate how the following fragment should be played, thereby introducing an additional layer of indeterminacy.

Zyklus for percussion comprises seventeen “periods” contained on spiral-bound pages. The performer is given the freedom to begin on any page, however, pages have to be played successively, ending with a repetition of the first stroke sounded at the beginning of the performance. The score allows the performer to read the notation in any direction, tempo, or orientation. In most of the periods, the performer can choose between different notation paths, introducing an additional layer of indeterminacy to a performance. Due to the arrangement of the instruments on the stage and deliberate notation choices in periods, the performer’s physical movements reflect the circular form of the piece, giving the audience an indication of the composer’s intentions. The identity of the score is preserved through the structure that defines the same triangular dynamic envelope for nine percussion timbres over seventeen periods. However, the starting point of each envelope is different for each timbre, creating an ever-evolving sound texture during a performance.

Due to the ambiguity surrounding the term “open form,” it would be useful, within the scope of this thesis, to narrow the discussion to “open form scores” that explicitly indicate the composer’s intention to delegate decision-making agency to the performance participants regarding musical event timing, ordering, or interpretation. In this context, an “open form score performance” becomes a collaboration between the composer and the performers, resulting in a real-time composition realisation driven by distributed decision-making.

In all historical examples mentioned above, the score notation is always static and predetermined. However, computer technology advances in the 21st century have introduced new composition and performance paradigms. In real-time algorithmic compositions, such as Richard Hoadley’s *Calder’s Violin* (2011), the music material decision-making is shared between the composer and the software algorithm. Although performers can practise this type of real-time notation during rehearsals, the actual notation content may differ in each performance. In the case of *Calder’s Violin*, the performer reacts to dynamically generated notation and digital audio in real-time, attempting the best effort performance that can be described as a combination of sight-reading and improvisation. The timing in *Calder’s Violin* is not explicitly defined and depends on the performer’s interpretation, as well as the rate of the notation change and the digital audio output generated by the algorithmic engine.

Various animated digital score visualisation implementations have enabled more precise graphic notation control during a performance. Cat Hope’s opera *Speechless* (2019a; 2019b),

performed using Decibel Score Player software (Hope et al. 2015), illustrates a large-scale graphic score composition performance scenario where the score position in time is synchronised for all performers. A static vertical line on performers' screens indicates the current score position, whilst the graphic notation scrolls below the position line. Performers execute the notation when the graphic notation elements reach the vertical position line. This approach explores Brown's "conceptually mobile" interpretational freedom whilst retaining some form of compositional control over temporal sound textures and rhythmic structures, even in scores written for large-scale ensembles.

As described above, computer networks can effectively distribute computer-generated or open form scores, granting musicians interpretive agency. The ZScore portfolio pieces take this concept further, offering broader decision-making agency to all performance participants, as detailed in Chapter 2.

1.3 Time, Action, And Sound Notation

This chapter explores the works of several composers whose composition methods and notation styles have influenced my composition practice. Networked portfolio scores build upon and further extend the described notation styles.

As described in Figure 1.1, sound and movement are deeply embedded in the history of human communication. Physical actions can be used as a communication device both in a direct visual sense or as an intermediary step causing the required information flow, such as the production of sound. It is not surprising that the oldest known music notation, clay tablets dating back to the Old Babylonian period (ca. 2000 - 1700 BCE) (The Schoyen Collection 2023), contained cuneiform tablature notation describing physical actions needed to produce a required sound.

In contrast, the common practice notation (CPN) used in Western classical music focuses on the definition of the resultant sound properties, such as its pitch and duration, leaving any decisions regarding the actions required to produce the notated sound to the discretion of performers. In addition, CPN notation may also contain instructions for particular playing techniques, such as vibrato, *col legno*, or *sul pont*, which may be interpreted as action notations. Traditions rely on the performer's understanding of the style, playing techniques, and aesthetics to complete the required information gaps.

In the second half of the 20th century, a number of composers aligned with the principles of modernism found the semantic constraints of common practice notation overly restraining in their pursuit of musical aesthetics ideals. The examples below illustrate the evolution of the notation for string instruments, however, the outlined principles can be applied to any instrumentation.

In his composition *Pression* for one Cellist (1969), Helmut Lachenmann rejects the conventions of common practice notation. His "musique concrète instrumentale" compositional aesthetics demands that "sounds are experienced as the immediate results of their production rather

than mediated by a historically loaded space of listening conventions and metaphorical meaning” (Utz 2017). The resulting notation approach, illustrated in Figure 1.4, describes physical actions to be performed by the cellist rather than the resulting sound. This approach implies a more visceral relationship between the performer and the instrument, creating an aural experience that also invites the listener of a live performance to engage with the sound production method.

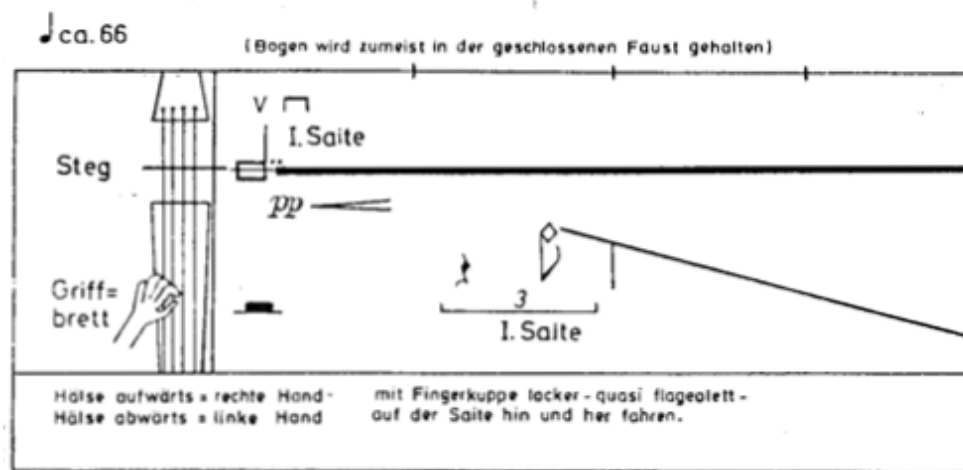


Figure 1.4: Helmut Lachenmann, *Pression* for one Cellist, score excerpt

The graphic clef resembling the cello’s shape depicted in Figure 1.4 indicates the approximate location of notated actions. The actions for the left and right hand are notated independently and can, therefore, produce multiple distinct sounds. The orientation of the music note stems indicates which hand should be used to execute the notated action (stem up -> right hand (bow), stem down -> left hand). This score is an example of a proportional notation system, where the horizontal axis represents the passage of time. In *Pression*, the distance between notches on the top staff corresponds to a quarter-note duration.

In *Time and Motion Study II* (1973-76) for cellist and live electronics (Figure 1.5), Brian Ferneyhough uses mixed notation consisting of common practice notation and graphic staves describing actions required to control various electronic components. The cello notation is split into left and right hand staves, allowing for independent hand movements. Here, the common practice notation describes both the resulting sound and the physical hand movement. The resulting sound is a mixture of the pitched and unpitched sounds produced by physical actions. The two graphic staves above and below the hand staves are for the foot pedals used by the cellist. Additionally, the cellist performs from the “Voice” stave containing pitched and unpitched sound definitions, making a total of five different staves dedicated to a single performer. The bottom three graphic staves are used by the performance assistants controlling the tape delays and the tape deck. Graphic staves indicate the approximate positions of the electronic controllers within the given boundary values.

It is important to note that the complexity of the notation in Ferneyhough’s works is the outcome of the aesthetic intentions driving the applied compositional process. In the *Time*

Figure 1.5: Brian Ferneyhough, *Time and Motion Study II*, score excerpt

and *Motion Study II* score preface, Ferneyhough stresses the importance of not stripping out “alternative layers of import” from the effect of the work or treating it casually as a piece of “music theatre”. The complexity of the notation conveys “fundamental musical processes of growth and decay which underpin and condition the entire gestural structure”.

In this piece, Ferneyhough explores the nature of time and memory through a recall of the “production process” by having the cello’s audio output captured on tape and replayed in a transformed way through delay machines, creating a sense of distortion and fragmentation of the remembered material. Whilst the cellist is firmly focused on the interpretation of the highly complex material, assistants are occupied with transforming, selecting, and reorganising the output of the “production process”. Despite Ferneyhough’s instructions to players not to treat his work as a piece of “music theatre”, for an audience member, it is hard not be captivated by the theatrical nature of the *Time and Motion Study II* ensemble performance.

The scores mentioned above also challenge traditional notions of musical time. Utz concludes that these new compositional methods require a fresh approach to musical analysis (Utz 2017).

His “morphosyntactic” approach focuses on the relationships between musical elements to understand how performers’ choices of tempo, dynamics, and articulation shape the experience of time. Utz identified three categories of form-building time-space concepts: “spatial time”, “processual time”, and “presentist time”. The analysis of multiple performances of the same scores incorporates combinations and interactions between these three categories.

For example, *Pression* can be characterised by a sense of spatiality (“spatial time”) as gestural movements on different parts of the cello are used to create key sound objects that anchor the placement of other events in time. Utz’s analysis identifies distinct sound events and relates them to one another in a hierarchical space, where the most salient events form the top layer. This technique understands the sound structure to be a “hierarchy of saliences”. *Time and Motion Study II*, on the other hand, may be best described with the processual model (“processual time”), where each sound event is presented as part of a large transformative chain over the duration of the entire work. The “presentist time” model can be applied to Iannis Xenakis’s *Nomos Alpha* for Cello, which comprises six parts split into two paths (A and B). For Path A, Xenakis uses eight “macroscopic” sound complexes organised through the theory of groups, resulting in fragmented and energetic music material. Path B contains contrasting long and quiet material that serves as intermezzi or ritornellos between the sections of Path A. The performer might choose either to accentuate the aspect of discontinuity and disruption (“presentist time”) or to establish continuities between the isolated events, allowing the music to unfold in a continuous stream of events (“processual time”).

In his *String Quartet No.3* (1982-84), Klaus Karl Hübler takes the idea of notated action separation further by splitting bow technique parameters, active strings, and left hand position notation into separate staves (Figure 1.6). The explanation of the staff layout in Hübler’s *String Quartet No.3* is shown in Figure 1.7.

Similarly to Ferneyhough, Hübler breaks the embodied integration of left and right hand string playing techniques by having separate notation streams that produce unusual timbral and rhythmical modulations. Hübler also employs novel tablature notation to achieve certain non-tempered pitches. In the example shown in Figure 1.8, the player is asked to apply 3rd position fingering in the 8th position. The tablature defines the width of a position by specifying the source position number (III) and finger distance, where dashes and dots signify fingers and empty semitonal spaces, respectively. The horizontal lines in Figure 1.8 indicate the relative finger movement during a glissando into the 1st position width. This notation approach reflects the composer’s intention to achieve exact passing non-tempered sounds by precisely notating actions that need to be taken by players.

Aaron Cassidy takes the tablature notation method further in his *Second String Quartet* (2009-10) (Cassidy 2020) by exclusively notating sound-producing actions in structured graphic staves (Figure 1.9). A significant shift in the notational aesthetics has been achieved by the utilisation of graphic design software, with its refinement and precision, as well as the introduction of colour. The staff structure in Cassidy’s *Second String Quartet* is based on a string instrument’s



Figure 1.6: Klaus K. Hübler, *String Quartet No.3*, Violin 2, page 8, score excerpt

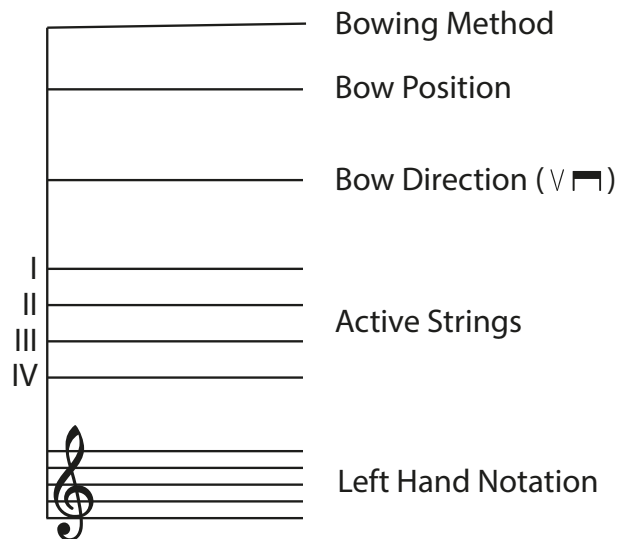


Figure 1.7: Klaus K. Hübler, *String Quartet No.3*, stave layout

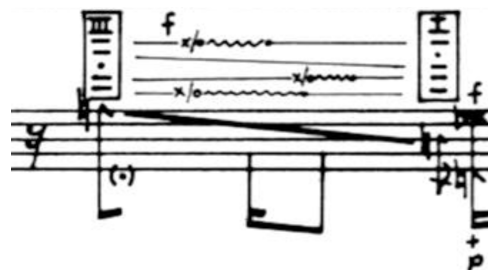


Figure 1.8: Klaus K. Hübler, *String Quartet No.3*, tablature notation

physical layout as shown in Figure 1.10.

Left hand movement is displayed in black. Similarly to Hübler, Cassidy utilises tablature

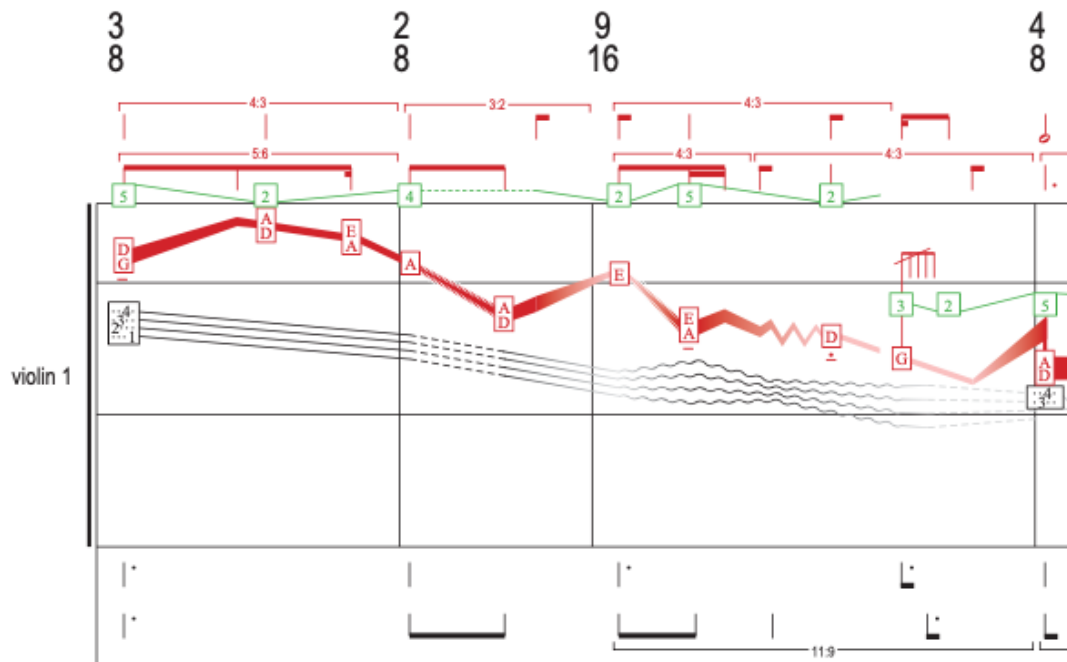


Figure 1.9: Aaron Cassidy, *Second String Quartet*, Violin 1, score excerpt

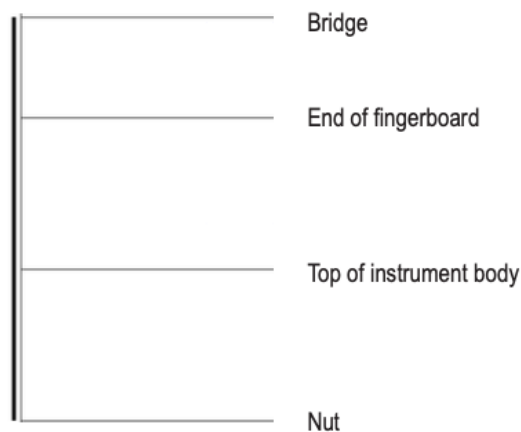


Figure 1.10: Aaron Cassidy, *Second String Quartet*, stave layout

notation indicating the position width and fingering. The movement line opacity and type indicate finger pressure intensity, or a playing technique such as vibrato or tremolo. Right hand motion is notated primarily in red. The graphical location of the string letter name and the connecting lines indicate the position of the contact point between the bow and the string. The width and opacity of the red line indicates bow pressure intensity whilst the purple colour is used for the col legno playing technique.

Green indicates the bow's direction (up/down) and speed. The bow is divided roughly into five equal "zones", with 1 towards the tip and 5 at the frog. Green boxed numbers indicate the position of bow zone contact based on this division. Musical dynamics notation is only provided for pizzicato attacks, as the dynamics of bowed material depends on the interaction of bow speed and bow pressure.

Left and right hand movements are rhythmically independent. Right hand rhythms are notated in red above the staff, whilst left hand rhythms are notated in black below the staff. There could be as many as four independent rhythm markings at any one time, referring to various bow and finger movements. The positioning of the rhythmical layers can be inconsistent and might include some guessing on the interpreter's part as to which playing technique the markings refer to.

This notation type, with its fluid hand movements and rapid changes in bow speed and pressure, results in unstable transient non-tempered pitch content, producing a wide range of sound dynamics from airy whispers to grinding noise. My research aims to continue development of the notation techniques and resulting sound aesthetics detailed above.

1.4 Music Composition In The Smartphone Era

This chapter contextualises my research within contemporary culture by examining the integration of digital technology into daily life and its impact on music-making processes. The term "Smartphone Era" refers to the period that began with the 2007 launch of the iPhone and extends to the present day.

Music-making, like any other continuous process, is inherently temporal in nature. Furthermore, it is inextricably linked to the time and culture from which it emerges. Authentic art-making, in my view, is akin to holding up a mirror to the arrow of time, reflecting an individual consciousness embedded in a surrounding human culture and formulating an expression that communicates as yet undiscovered valuable aspects of the sensed reflection. Unlike an optical mirror, which reflects a three-dimensional physical space, the imaginary art mirror holder perceives a reflection of the present state of mind and culture in the foreground, with all previous experiences and knowledge acquired through time, chained in the background.

The ever-increasing presence of digital technology in today's world is undoubtedly having a significant impact on society. As Katherine Hayles argues (Hayles 1999), the posthuman condition is not some distant future state but an ongoing process characterised by the constantly evolving relationship between humans and technology. This transformation brought about by digital technology has not only changed the way we think and communicate but has also given rise to new forms of social organisation and identity.

Digital technology has become ubiquitous in music-making practice today. Tanaka argues that "all music today is some form of music-computer interaction" (Tanaka 2019). The symbiotic relationship between music performers and technology, however, is not a recent phenomenon. Nijs and Lessaffre (2013) analyse the depth of embodied interactions between musicians and their instruments and conclude that musical instruments, whether acoustic, electronic, or digital, act as an extension of the musicians' bodies and minds. The human-nonhuman entanglement is explored in Ferneyhough's *Time and Motion Study II* (Figure 1.5), where an organic body (cellist) intersects with technologies both mechanical (cello) and electronic (microphones, foot

pedals, tapes, etc.).

Just as acoustic instruments can be seen as an augmentation of the human body:

“The musical instrument is a prosthetic augmentation of the human body, enabling the body to exceed itself (to sound faster, higher, louder than any voice, and to enable the individual to do so often in multiple parts simultaneously)” (Johnson 2015, p. 142),

Similarly, computer networking technology can be treated as an extension of human communication capabilities in music-making, enabling us to exchange larger amounts of diverse information, with higher frequency, lower latency, and the ability to do so often with heterogeneous participant types simultaneously. This enhanced communication allows for the exchanging of data and events that can trigger behaviours enacted by both human and digital actors, thus further expanding the boundaries of human-nonhuman interaction.

Pinchot et al. (2011) describe how mobile phones have become an integral part of our daily lives, changing the way we communicate and interact with others. They argue that mobile phones have significant implications for our understanding of communication, identity, and creativity, thus providing new opportunities for embodied interaction and expressive capacity.

These developments have been exploited by a new generation of digital musicians, leading to the formation of several ensembles experimenting with mobile phone utilisation in music-making, such as the Stamford Mobile Phone Orchestra (Wang et al. 2014). Swedish composer Anders Lind (2020b) has created several compositions for mixed ensemble performances that include the Mobile Phone Orchestra (2020a). In these performances, mobile phones are mainly used as sound sources, acting as an extension of the traditional orchestral instrument concept. Due to the limited power of the built-in speakers, the sound produced by the mobile phones is usually amplified. The technical implementation of the software used in Lind’s scores does not allow for direct communication between mobile phone performers and acoustic performers. In this case, mobile phone performers observe a score displayed on a large screen and react to it by triggering sound samples on their mobile devices. Conceptually, the audience’s role in this scenario is still regarded as separate from the mobile phone performers, who actively participate in the performance.

When composing for a combination of acoustic instruments and mobile phones, it is important to consider the limitations of mobile phone speakers, such as their relatively low power output (1 - 3 W) and narrow frequency range (~500 Hz - 10 kHz). Despite these limitations, an average mobile phone can still achieve a loudness of 70 - 77 dB SPL, which is comparable to the sound produced by a violin played *mezzoforte* (maximum violin loudness 75 - 95 dB). It is worth noting that doubling the number of mobile phones in the ensemble will only result in a modest increase of 3 dB in overall loudness.

Barbara Lüneburg’s research project on Gamified Audiovisual Performance and Performance Practice (GAPPP) (Lüneburg 2018) explores participatory culture by involving an online audi-

ence familiar with gaming and social media technology in the making of a multimedia artwork. Lüneburg (2017) observes that, in order to enable performative involvement over an interactive system, a designer needs to create meaningfulness with regard to:

- Technical skills. The work allows performers to gain or enhance their technical capabilities and thus heightens their sense of agency.
- Creative strategies and goals. The work's rules, strategies, and objectives should be clear and recognizable, granting performers agency to influence the work musically, visually, strategically, or content-wise.
- Musical objectives. The system and the work should offer options for making musical decisions that make traceable sense and are satisfying to performers, possibly presenting unexpected but challenging contingencies.
- Sharing artistic experiences with the audience. The system and the work allow for performative decision-making and actions that let performers transmit the artistic experience on a social and artistic-communicative level in cognitive, sensor-motoric, or emotional ways.

Lüneburg's list of requirements for meaningful performative involvement provides valuable guidelines for developing participatory system interfaces aimed at audiences familiar with mobile phone technologies, regardless of their exposure to contemporary music performance practices.

Digital technology and the cultural changes brought about by its ubiquitous presence have inevitably impacted the nature of a music score. Craig Vear (2019) captures the recent shifts in musicianship, creativity, and innovation resulting from the transformation of music-making practices through engagement with digital music scoring systems, which he refers to as "the digital score". Vear's theoretical framework for understanding the emergence of the digital score defines fundamental principles governing its purpose and function:

"The core purpose of a digital score is a technically mediated communication interface between the creativity of a composer, the creativity of a performer and the creative mind of the listener.

The core function of this communication interface is to represent the ideas that happen inside the mind of the composer, using digital technology in such a way that these ideas are capable of being translated into sound during performance through the technique and creative interpretation of a performer (human or machine)." (Vear 2019, p. 19),

Based on these fundamental principles, Vear defines six features that characterise the nature of the digital score (Vear 2019, p. 30):

- A digital score is a technically mediated communications interface that enhances how ideas in music can be represented.
- A digital score is a technically mediated pathway for a musician (human or machine) to navigate within sound during the performance.

- A digital score is a hardware-software combination that can support and enhance the connectivity of people, sound, space and score.
- A digital score allows compositions to be defined by their interactivity.
- A digital score can augment performance techniques that lead to invention and creativity within the parameters of active composition (especially improvisation and open/distributed compositional forms).
- A digital score is a technological space for creative invention.

In a networked environment, the core purpose of the digital score can be expanded to not only mediate communications but also to enable active participation and decision-making agency for all performance participants. Creative ideas in the minds of composers, musicians, and audiences can be translated into sound, visual, or gestural representations mediated by the digital score.

Identifying the origins of creativity in human-computer artistic practice is becoming a non-trivial issue. Musical metacreation (MuMe) is a field that studies the partial or complete automation of musical tasks (Pasquier et al. 2017). In the digital age, the autonomous tools that can perform these tasks include Artificial Intelligence (AI) and Multi-Agent Systems (MAS) (Tatar and Pasquier 2018). MAS are distributed systems that run on a network, where each software agent is an autonomous entity containing perception and action abilities.

One of the frameworks that implements MAS architecture is the Musebot project (Bown et al. 2015). It offers a platform for interactive live performances involving human performers and multiple musical agents over a network. Musebots are autonomous agents capable of performing music-making tasks and collaborating with other connected musebots over a network. Communication is facilitated by the “musebot conductor”, a standalone software coordinating a collection of musebot agents on a local network, known as a “musebot ensemble”. Whilst this research does not include AI and MAS, future work could explore integrating musebots within the ZScore system, opening a new range of possibilities for human-computer networked music-making.

This study focuses on the exploration of human creativity within an enhanced human-computer performance environment. The ZScore system and the accompanying portfolio aim to expand Vear’s definition of a digital score by providing decision-making agency and meaningful performative involvement to all performance participants, as advocated by Lüneburg.

1.5 Networked Music-making

This chapter explores the technical aspects of computer networks relevant to music-making systems. It examines both the advantages and potential drawbacks associated with various network solutions and protocols. This analysis informed the design and implementation of the ZScore system.

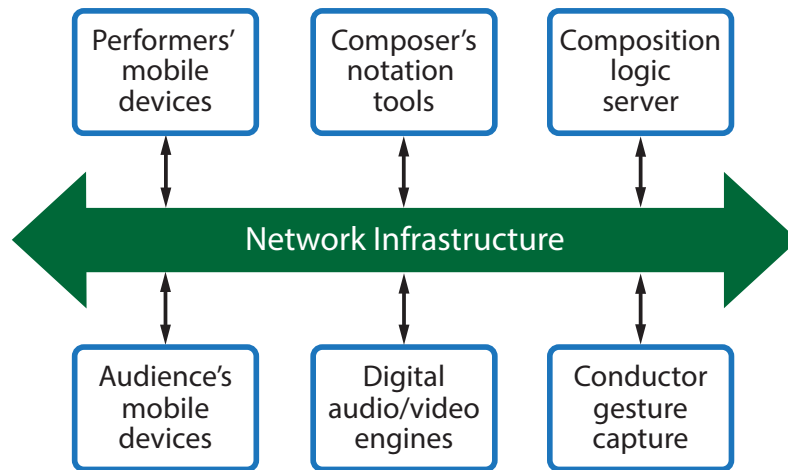


Figure 1.11: A networked music-making environment

Networked systems allow users to communicate and coordinate actions by passing messages over a computer network. Most network systems, including the Internet and Local Area Network (LAN) (Clark et al. 1978), are based on the Open Systems Interconnection model (Costa 1998) which ensures connectivity between heterogeneous clients running on different hardware and software implementations. Modern high-throughput, low-latency messaging systems are able to process thousands of messages per second with sub millisecond latencies. Network messages can both transfer large quantities of data from one location to another and carry events that trigger actions at different target nodes. Event-driven systems usually rely on a set of rules that define the type and timing of actions that need to be executed upon event receipt. Considering that event-driven networked systems provide both fast communications between different actors and decision-making functionality, they are well-suited to be used in a music-making activity. Networked nodes can produce audio, visualise notation, or execute algorithms with latencies imperceptible to humans (below 10 milliseconds).

Networked music-making systems allow all participant types – musicians, audience, composers, conductors, and various digital audio and video engines – to connect and exchange data and events in real-time² (Figure 1.11).

Unlike traditional one-directional communication flows (Figure 1.2), networked systems allow for multi-directional communication between all participants (Figure 1.12). Any participant in a networked system can be assigned sound production or decision-making ability, thereby blurring the boundaries between traditional music-making roles. Furthermore, networking technology allows for unconstrained control of the improvisation decision-making dial (Figure 1.3) as required by the context of a performed piece of music.

The main network communication models that define how data is transmitted and received between connected devices are illustrated in Figure 1.13. Unicast is the simplest and most common type of computer communication, where a single sender distributes data to a specific

²The term “real-time” here does not imply a guaranteed system response but rather refers to close-to-instantaneous communications.

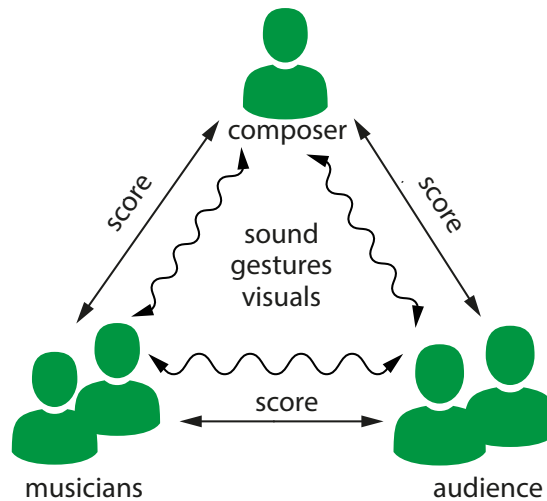


Figure 1.12: Multi-directional communication in networked music performance

recipient. In this one-to-one or point-to-point communication, a source transmits a copy of the data to each destination. In a scenario where a single sender has to distribute data to many different destinations, this model creates a significant load on the sender side and the network.

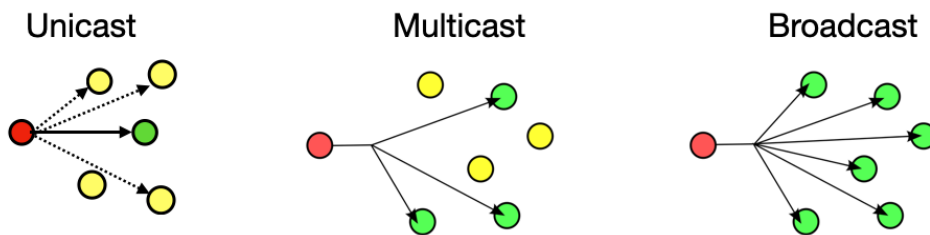


Figure 1.13: Network communication models

The multicast model allows a sender to transmit data to a group of recipients simultaneously. In this one-to-many communication, a sender sends the message only once, and the network infrastructure itself duplicates and distributes the data to all group members. This model is more efficient and reduces network congestion when a single sender produces messages for several consumers. It is particularly effective when different participant groups, such as orchestra sections, require different message content.

In the broadcast model, a single sender sends data to all devices within a network. Whilst this one-to-all model may seem to be an appealing solution when a message needs to be sent to all connected devices, it requires caution. Serious issues, such as a “broadcast storm,” may occur when a large number of packets are sent through a network in a short period of time, overwhelming the network and disrupting all communications.

Network communication protocols define a system of rules that govern how data is transmitted, received, and processed in a computer network. The Transmission Control Protocol (TCP) (Cerf and Kahn 1974) establishes a reliable, one-to-one connection between a sender and a receiver. Internet browser connectivity is mostly delivered over the TCP/IP protocol suite,

which includes HTTP (Berners-Lee 1991) request-response communication and fully-duplex WebSocket (W3C 2009) connections that enable bi-directional communication between clients and servers. This protocol ensures reliable messaging by requiring the receiver to acknowledge (ACK) the receipt of each packet. However, the overhead of socket connections and packet acknowledgements can impact data transmission rates, resulting in reduced network throughput.

The Maximum Transmission Unit (MTU) in TCP communications, typically set by network card manufacturers at 1500 bytes for Ethernet protocol, determines the largest packet size that can be sent over a network. Larger MTU sizes may reduce latency as more data can be packed into a single message, whilst smaller MTU sizes can minimise the impact of packet loss. The default value is usually sufficient, however, adjusting the MTU can be explored if the music-making system experiences latency or packet loss issues. On an application level, latency-critical data structures intended for network transfer should be designed with this limitation in mind to avoid unnecessary data fragmentation during transmission. For example, large strings or files should be avoided in latency-sensitive communications.

Unlike TCP/IP, the User Datagram Protocol (UDP) (D. P. Reed and Postel 1980) is a lightweight, connectionless protocol that does not provide reliable packet delivery. The UDP is suitable for applications where fast data transfer is prioritised over guaranteed packet delivery. It is commonly used as the transport layer protocol in multicast implementations. The maximum UDP packet size is 64 KB, so any larger data structure may be truncated during transmission and should be split into multiple messages. For instance, large linked lists containing score data might be entirely lost during UDP transfer, therefore, they should be divided into smaller sublists, sent in multiple messages, and reassembled on the receiving node. ZScore keeps track of all messages exceeding the configurable size limit and raises an alert if the limit is breached. Offending data structures can then be refactored to adhere to the size limitations.

To address potential UDP message loss, additional mechanisms can be implemented on top of the UDP protocol. This can involve using Reliable UDP (RUDP) (Bova et al. 1999) libraries or implementing an application-level mechanism that tracks message order and requests replays for missed packets. Unlike TCP, which requires acknowledgment for each packet, reliable UDP mechanisms usually employ negative acknowledgment (NACK) to inform the sender when a specific packet is not received, triggering a replay.

Digital devices are usually connected to a computer network either via a wired Ethernet connection or wirelessly through Wi-Fi technology. Wired Ethernet connections offer speed and reliability, whilst Wi-Fi offers convenience and scalability. All modern mobile devices support at least one version of Wi-Fi connectivity. On the negative side, Wi-Fi signals can be affected by physical obstructions, distance, and other devices operating on the same frequency range. These factors can lead to signal degradation and reduced network performance. Signal interference can cause packet loss and message latencies measured in hundreds of milliseconds. Additionally, Wi-Fi can be vulnerable to unauthorised access if not properly secured, especially in publicly available networks.

Networked performance systems in public venues should be designed to protect core components against malicious attacks during concerts, such as unauthorised access or denial of service. This may entail the installation of network firewalls that segregate public and private local networks, along with encryption and robust authentication mechanisms for the protected internal segments of a network.

There are several types of networked system architectures that determine how network clients, such as the musicians' tablets, the audiences' mobile devices, or audio and video digital engines, connect to the rest of the system. Figure 1.14 illustrates basic network architectures that can be utilised in music-making systems.

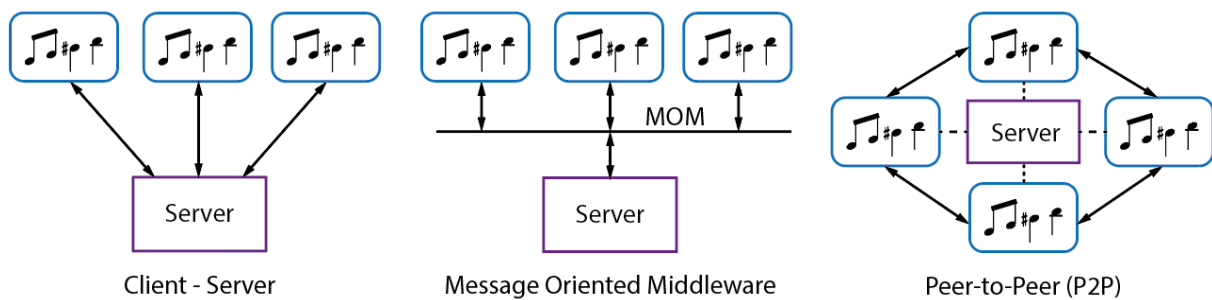


Figure 1.14: Network application architectures

The most prevalent solution by far is the standard Client-Server architecture, where all networked devices connect directly to a server responsible for data processing and distribution. Due to the widespread adoption of web technologies, this method often utilises standard web connectivity protocols such as HTTP or WebSocket. Another useful option for Client-Server communication is Server-sent Events (SSE) technology (2006). SSE is a server push technology enabling a client to receive automatic updates from a server via a HTTP connection. As HTTP port access is usually allowed, this technology enables data transfer through almost any network firewall.

Most modern mobile devices come with at least one web browser installed by default, which can act as a network client in music-making. The recent adoption of HTML5 (2008) and various JavaScript (Eich 1995) frameworks makes them well-suited for the development of dynamic notation interfaces. Additionally, there are several proven and robust web server implementations available for all operating systems, including NginX (2004), Apache (1995), and Microsoft IIS (1995). Another popular solution amongst web developers is Node.js (2009), which offers server-side JavaScript development and asynchronous, non-blocking server-side processing, making it suitable for music-making applications.

Connectivity in the Client-Server architecture can also be established over UDP Unicast or Multicast on local area networks. However, if network traffic needs to pass through a firewall, additional configuration may be required to ensure the uninterrupted flow of UDP packets.

Depending on the number of connected clients, the Client-Server architecture may impose a significant load on the server-side. Additionally, in the simple scenario illustrated in Fig-

ure 1.14, the server becomes a single point of failure. Mission-critical applications that require high-throughput, low-latency processing, such as financial instrument pricing and trading applications, often deploy robust Message Oriented Middleware (MOM) as a messaging layer between all connected devices. MOM provides a scalable and fault-tolerant solution by decoupling the communication between clients and servers. It allows for asynchronous message passing, which reduces the reliance on real-time connections and minimises the impact of single points of failure.

Whilst many MOM solutions are tailored for enterprise-level implementations, there are several open-source, relatively lightweight, multiplatform, fast, and reliable messaging implementations available. Examples include ZeroMQ (2007) and Aeron (2014). Both messaging systems are capable of delivering millions of messages per second with sub-millisecond latencies³. One drawback of MOM implementations is the requirement for proprietary software installation on the client-side. In the case of music-making systems, for instance, all audience members would need to install a dedicated app on their mobile devices to participate in a performance.

Peer-to-Peer (P2P) is a decentralised form of communication where devices can connect and interact with each other without the need for a central server. In music-making applications, however, the most likely scenario would be for clients to first connect to a server and obtain the required score data before establishing P2P connections and directly exchanging data and events with other peers during a performance.

Web Real-Time Communication (WebRTC) (2011) provides a framework for browser-based applications to establish P2P connections, enabling the transmission of audio, video, or any other data required for a composition performance. Although most major browsers support WebRTC, implementations can still be inconsistent. WebRTC has not been utilised in this project. However, as part of future development, I plan to investigate the possibility of UDP messaging over WebRTC, as UDP and multicasting are currently not supported by web browsers.

Open Sound Control (OSC) (2002) is a protocol specifically designed for exchanging messages and data related to sound, music, and multimedia applications. It was originally intended as an alternative to the MIDI (1983) standard, offering higher resolution and a richer parameter space for communications between electronic music instruments, such as synthesisers. However, the protocol proved to be suitable for much wider scoped platform-independent implementations, enabling different software and hardware devices to communicate over a computer network and control one another in real-time. The majority of OSC implementations are built on top of UDP, inheriting both the benefits and drawbacks of UDP messaging. Connected devices can send and receive messages containing specific commands, parameters, and data related to sound and multimedia control.

OSC protocol provides a simple scheduling mechanism. Single messages are executed as soon as possible after they are received, whilst the message bundle execution time can be specified

³Sources: 100GbE Tests with ZeroMQ (iMatix 2021), Aeron latency tests (Pirogov 2017)

in the OSC Time Tag property. If the Time Tag value is equal to or before the current time then the server invokes message methods immediately. Otherwise, the server schedules message execution at the time specified in the Time Tag. However, OSC does not provide or specify a mechanism for clock synchronisation between different network nodes. Time synchronisation is crucial in music-making applications that require accurate sound production or notation timing across all network nodes. OSC relies on lower-level computer network time synchronisation mechanisms, such as the widely available Network Time Protocol (NTP) (Mills 1985), which has become a de facto standard for network time synchronisation. Lévesque and Tipper (Lévesque and Tipper 2016) provide a detailed description of network clock synchronisation methods and comparison of standard protocols in their survey of existing solutions.

NTP is a hierarchical solution where network nodes obtain accurate time information from a server higher up in the hierarchy. It typically operates through Internet connectivity, with top-level servers utilising atomic clocks or GPS signals for high accuracy. In local area networks (LANs) without Internet connectivity, such as a small network inside a music performance venue, at least one computer must run an NTP server that acts as a master for time synchronisation within the network. The drawback of NTP is that inadequate network topology and data congestion can lead to delays of 100ms or more, making it unsuitable for highly time-sensitive applications like music performance.

Precision Time Protocol (PTP) (2002) is an alternative high-precision time synchronisation protocol designed for local area networks. PTP can achieve sub-microsecond synchronisation accuracy by utilising hardware timestamps and sophisticated synchronisation algorithms. Similar to NTP, one of the computers on the network needs to run a PTP master instance that other slave nodes on the network can use to obtain accurate time. However, PTP support is hardware-dependent and may not be available by default. It may require installation and configuration on all participating network nodes. Alternatively, PTP-enabled hardware routers and switches can be used on the LAN as master clocks, providing a centralised and hardware-based time synchronisation solution.

For Internet-wide performance, the most accurate master clock that can be used is the GPS time signal, which has a theoretical accuracy of 14 nanoseconds. An example of music system synchronisation over a GPS signal is The Global Metronome project (Oda and Fiebrink 2016). It demonstrated that the combination of GPS for the master clock and NTP for LAN synchronisation can achieve sub-millisecond network node clock offsets. The main issue with The Global Metronome is that it requires access to a GPS signal, which means having a clear view of the sky – this is typically unavailable in music venues. A possible solution is to place The Global Metronome externally, connect it to a LAN within a performance venue using Ethernet cables, and use NTP to synchronise nodes on the network. This approach can provide highly accurate time synchronisation for music-making applications within the venue whilst leveraging the precision of the GPS signal from an external source.

An alternative high-precision method for clock synchronisation over a network is Reference

Broadcast Synchronisation (RBS) (Elson et al. 2003). Instead of sending a timestamp in a synchronisation packet, it allows nodes to use the packet's arrival time as a reference point for clock synchronisation. The synchronisation packets are broadcast to all network nodes, which then exchange the receipt time with local neighbours to calculate their phase offset and estimate clock skew. The main drawback of this approach is that applications that do not support broadcast, such as web browsers, can not be synchronised this way. Furthermore, RBS inherits possible issues related to message broadcasting on congested networks as described above.

In many cases, it may be more practical to synchronise notation and event execution over a network using tempo-relative time rather than absolute time. This approach, similar to the MIDI beat clock implementation (Malouin 2020), relies on the master application sending synchronisation events at regular intervals (e.g., every fraction of a beat). These events allow participating nodes to establish their own internal tempo-relative positions within the musical timeline. Additionally, the local system clocks of network nodes can be used for more granular scheduling and synchronisation within the established tempo framework. To achieve acceptable synchronisation accuracy, it may be necessary to track the network latency per node and make event timing adjustments based on the individual node latency. This approach ensures that the event timings align as closely as possible across the network, compensating for any variations in network jitter and latency. ZScore utilises a proprietary tempo-relative node synchronisation technique, as detailed in Chapters 2, 3.2.1.3, and 3.2.1.4.

The exploration of various networking aspects in this chapter directly influenced the development of the ZScore system. For instance, the system's client-server architecture facilitates both TCP and UDP messaging for flexible communication, the message payloads are optimised for the standard MTU size to minimise network fragmentation, and the message size never exceed the maximum UDP packet size to avoid potential packet loss. Some explored concepts, such as leveraging WebRTC for UDP multicast to web browsers, form the basis for planned future development.

1.6 Existing Solutions

In this chapter I explore the most prominent existing networked notation systems and related software used in the music-making domain. The development of the ZScore system was based on this analysis of the strengths and weaknesses of existing solutions.

A number of software solutions capable of dynamically rendering music scores distributed over a local or wide area network have been developed in the last two decades. Figures 1.15 and 1.16 list software applications that have had the most impact in the field of networked music notation during this period. Individual applications are discussed further in this section.

Some composers and laptop orchestras in particular develop proprietary composition and performance software in programming environments, such as Max (Puckette and Zicarelli 1990), Pure Data (Puckette 1996), OpenMusic (IRCAM 1998), SuperCollider (McCartney

Name	Platform/ Server	Notation Client	Comment	Reference
MaxScore	Max, Ableton Live	Java (mxj)	Mixed notation authoring	Didkovsky and Hajdu 2008
Quintet.net	Max	MaxScore	Up to five notation clients	Hajdu 2005
Drawsocket	Node.js, Node for Max	Web Browser	JavaScript client, MaxScore notation	Gottfried and Hajdu 2019
INScore	Standalone/ Any	Standalone	Time-Space object mapping	Fober et al. 2013
INScore Web	Standalone/ Any	Web Browser	WebAssembly client	Fober et al. 2021
Decibel ScorePlayer	Standalone	Standalone	iOS app only, graphic scores	Hope et al. 2015
Bach	Max	Max	Open Source API	Agostini and Ghisi 2012
dfscore	Node.js	Web Browser	Comprovisation	Constanzo 2014
Comprovisador	Max	Bach	Algorithmic dynamic notation	Louzeiro 2018
Score Viewer	Node.js	Web Browser	Polytemporal scores	Opstad 2022

Figure 1.15: Networked notation software summary

1996), Processing (Fry and Rea 2001), and the ChucK (Wang 2003) programming language. Whilst all notation applications share common high-level functional objectives, their internal data models, score rendering versatility, system dependencies, time synchronisation strategy, and modes of communication vary greatly. As a common feature, they all have a front end running on a computer or a mobile device capable of rendering dynamic music notation, and some kind of logic running either on a server (such as Max, Node.js, etc.), or directly on a client (Decibel ScorePlayer), that distributes data and events over a network.

Most of the solutions rely on the Open Sound Control (OSC) (2002) protocol for communications between network nodes. INScore supports OSC natively, whilst Max-based solutions use various built-in and third-party OSC implementations packaged as Max objects. Quintet.net additionally uses TCP protocol where reliable messaging is required to ensure the successful transmission of critical data and control messages.

The Odot framework, being a middleware-oriented messaging solution (MacCallum et al. 2015), is a welcome step towards network services abstraction and encapsulation. This approach hides internal system complexity and improves coding efficiency by exposing only necessary functionality. Developers can treat Odot as a “black box” responsible for network communication

Name	Platform/ Server	Comment	Reference
Odot	Max, Pd, OpenMusic	Aggregate data type, OSC encoding, expression language	MacCallum et al. 2015
Landini	SuperCollider	Client management, network time sync, message replay	Narveson and Trueman 2013

Figure 1.16: Middleware solutions for networked music-making

tasks, allowing them to focus on delivering music application logic. Odot wraps the OSC protocol and provides transcoding to JavaScript Object Notation (JSON) ([2001](#)), Scalable Vector Graphics (SVG) ([2001](#)), and S-Expressions ([1960](#)), as well as bindings to JavaScript and Lisp. This allows application developers a great deal of flexibility regarding the type of information they wish to distribute to network nodes. On the negative side, as with all “black box” solutions, developers lose fine grained control over the encapsulated logic. For example, SVG data transcoded to OSC can result in a large quantity of messages that can lead to network saturation, in which case, other more optimised solutions might be more efficient.

Landini (Narveson and Trueman [2013](#)) can also be classified as a form of messaging middleware as it creates an additional layer between music applications communicating over OSC. Landini implements a reliable, ordered message delivery protocol which detects packet loss and attempts recovery. Furthermore, it monitors network latency and applies OSC timing corrections for more accurate event synchronisation. However, this method introduces an additional hop and, therefore, additional latency in all communications between network nodes. A more targeted approach, intersecting only specific message types, might yield better results.

Most of the existing compositional tools are capable of rendering traditional symbolic notation. Support for graphical notation, custom symbols, staves, or extended performance techniques, is commonly achieved by layering SVG or raster graphics on top of symbolic notation. However, individual graphics file layering often does not provide enough accuracy and flexibility for complex score authoring compared to native SVG implementations. Anchoring fixed graphics on top of existing notation frequently results in inconsistent positioning and sizing, especially when switching between different client implementations (e.g., browser makes) and screen resolutions. This approach can also increase score transfer timings and lead to latency in notation view rendering.

The maximum number of parts allowed in a score is either restricted explicitly or by the available application memory. Delivering larger instrumentation, such as a full-sized orchestra, remains a significant challenge, especially when tight timing synchronisation is required. The most impressive results in terms of scalability have been achieved with Drawsocket for the performance in St. Pauli Elbe tunnel ([2019](#)), where 144 musicians spread out over 864 metres performed several live pieces.

Scores are typically composed offline using a proprietary data model specific to each notation

application. If needed, these scores can be converted to one of the common notation formats, such as GUIDO (Hoos and Hamel 1997), JMSL (Didkovsky and Burk 2004), or MusicXML (Good 2001), for sharing with other notation applications. Currently, there is no clear winner amongst competing symbolic notation formats, and each format has its strengths and limitations. Real-time notation generation and distribution are well-supported in modern networked music-making systems. However, communication between heterogeneous applications usually requires transcoding of the native data models to OSC or other similar formats on all participating nodes (James et al. 2017).

For time synchronisation between network nodes, notation applications typically either rely on the system clocks or regular heartbeats sent from the master node. Network Time Protocol (NTP) is used by default on most LANs for system clock synchronisation, but it can cause inaccuracies of up to 100 ms between computer clocks due to network latency and other factors. The application scheduling resolution, which defines the minimum time interval between two scheduled events, is normally defined in the milliseconds range.

Quintet.net, conceived and developed by Georg Hajdu (2005), is a seminal contribution to the field of networked music technology, offering comprehensive functionality such as notation authoring, real-time network distribution, dynamic notation rendering, integrated digital audio and video engines, as well as strategies for mitigating network jitter and latency. It consists of several units:

- Server unit: in charge of data distribution, state and connectivity management.
- Multifunctional Client: acting as a notation unit, MIDI and audio data input unit, and a sound synthesis unit.
- Conductor unit: in charge of up to five score parts, their settings, and text message management.
- Viewer unit: in charge of visual components.

Quintet.net runs on the Max platform and can be deployed either on a local network or over the Internet. The system has been utilised in various performance scenarios, including mixed electro-acoustic ensembles playing composed and partially improvised scores. For notation visualisation, Quintet.net utilises MaxScore, a Java (mxj) object running on the Max platform. MaxScore implements the Java Music Specification Language API (JMSL) (2004), and until recently, a JMSL licence was required for its use. Functionally and aesthetically, MaxScore is similar to other commercial notation authoring software like Sibelius (1993) or Finale (1988), but with the added advantage of Max integration. Utilising the Max platform brings multiple benefits due to its processing power, flexibility, extensive features, and maturity. However, Max is also commercial, closed-source software that creates dependencies on underlying libraries, inherits potential high demand for computer resources, and necessitates the inclusion of unnecessary extraneous features. As reflected in the name, Quintet.net only supports up to five parts and is therefore not suitable for large ensemble scores.

INScore (2013), created by Dominique Fober, is another well-designed and innovative application offering dynamic notation rendering over a network. It natively supports networking (OSC), SVG graphics, JavaScript, and Guido Music Description Language (GMN) (1997) for symbolic notation. INScore's unique feature is its simple and flexible time synchronisation method in two-dimensional graphic space, allowing any arbitrary object's position and size to be synchronised in time (Fober et al. 2010). Time coordinates can be expressed as a relative value linked to musical metre or an absolute time measured in minutes, seconds, and cents. INScore also can host several plugins, such as FAUST (2002) for real-time signal processing and synthesis, gesture follower, and Httpd (Web) server, and can therefore be used for rich multimedia performances. For these reasons, INScore was chosen to serve as the front end for notation rendering and audience score visualisation in early versions of ZScore.

Decibel Score Player (Hope et al. 2015), developed by Aaron Wyatt, is an iOS app originally made for the Decibel New Music Ensemble under the leadership of Cat Hope. The app allows for a synchronised performance of primarily graphic scores over a computer network. It is relatively lightweight and simpler to use in comparison to other networked notation solutions, as it does not require a dedicated server. All ScorePlayer instances discover and synchronise with each other using Apple's Bonjour (2002) zero-configuration networking. Whilst this approach offers a straightforward connectivity solution, it may lead to excessive network traffic and introduce control and security issues, as any malicious client could potentially take control and impact all other connected devices during a performance. Decibel ScorePlayer utilises a scrolling score paradigm where the static vertical line indicates the current position. It has been extensively utilised by the Decibel Ensemble and larger scale forces, as in Cat Hope's opera *Speechless* (2019a). The related macOS desktop application, Decibel Score Creator, enables authoring of the scores and related metadata required by the Decibel ScorePlayer.

Comprovisador (Louzeiro 2018) uses Bach (Agostini and Ghisi 2012) notation software, written for the Max platform, to visualise dynamic notation over a network. The system employs machine listening and algorithmic compositional procedures to enable improvised performances, where the improvising soloist triggers algorithmically created notation displayed on other musicians' screens. The conductor can modify the configuration of the compositional algorithms during a performance to adjust the system's output. The notation client also displays an animated position indicator in the form of a bouncing ball, akin to the word pointer used in karaoke applications. Comprovisador's use case is limited to the common practice notation available in Bach software.

Opstad (2022) developed proprietary software for polytemporal scores. In this solution, score functionality primarily operates within a web client. The Node.js server hosts static data, distributes basic transport commands, and provides clock synchronisation through the @ircam/sync JavaScript library (IRCAM 2018). The client logic is fully in charge of the score display, making this implementation unsuitable for use cases where score representation may be modified by the actions of other users or the server in real-time. The clock synchronisation

mechanism inherits JavaScript’s threading model issue, where any blocking of the event thread on the client or the server side may cause time drifts. ZScore supports polytemporality through multiple transports that can be associated with different instrument parts.

In recent years, there has been a growing trend towards web-based solutions in the field of networked music technology. A common feature in web notation implementations is the deployment of a WebSocket point-to-point connectivity between a browser and a server. Examples of such solutions include Drawsocket (Gottfried and Hajdu 2019) and dfscore (Constanzo 2014), both of which are JavaScript based implementations. The latest version of INScore (Fober et al. 2021) uses WebAssembly (Wasm) (2017), which allows native code to run within a web browser container. WebAssembly is a relatively new technology that is not yet widely supported by various browser makers. Its use is recommended only for browsers in which the specific application has been thoroughly tested. Jonathan Bell’s excellent paper (Bell 2021) provides a detailed comparison of different browser-based score systems.

Despite being a relatively new project, Drawsocket (2019), developed by Rama Gottfried, has already been widely adopted by the networked notation community. Drawsocket utilises the Odot “o.io” framework (Freed et al. 2014) to provide OSC API for various data formats, such as SVG and Cascading Style Sheets (CSS), within a web browser. It requires node.js or Node For Max (N4M) for server-side processing. Its original purpose was to render MaxScore notation in web browsers; however, it can be used for various multimedia browser visualisations. Drawsocket’s JavaScript notation client is relatively thin, containing only the logic required for Odot message processing and notation view rendering. The score state is held on the client-side, apart from the message archive that is stored on the server-side. The message archive is used to replay all client-specific messages if a client disconnects and reconnects at any point during a performance. The idea behind the latest project Symbolist (Gottfried 2022) is to provide dynamic notation authoring within a web browser. The combination of Symbolist and Drawsocket has the potential to provide support for integrated composition and performance, where all user front ends can run within web browsers.

Most of the existing network notation solutions attempt to replace and enhance the functionality of the traditional static paper scores used by trained musicians. However, the capabilities of the networked systems allow for the integration of any network-enabled device, including audience members’ mobile phones, digital audio and video engines, conductor or dancer gesture trackers, etc. Networked systems need to evolve to enable the authoring, hosting, and distribution of scores that incorporate notation and interactive features for all different devices and participant types.

1.7 Networked Notational Perspective

Whilst exploring the distinctive characteristics and possibilities of networked music notation, this chapter also acknowledges the increased complexity and technical expertise required to

realise the transformative potential of interactive, dynamic scores.

As argued by Bhagwati (2013b), music notations from different traditions tend to optimise written information by focusing on elements considered essential or repeatable within a performance context. Elements that are contingent or ubiquitous within a particular tradition tend to be omitted. For example, in Western classical music tradition, it was considered important to notate pitches and their durations, whilst the exact articulations or playing techniques, such as exact bow pressure and speed, were frequently not specified prior to the 20th century. In contrast, Chinese art-music tradition considered the notation of pitches and associated playing techniques important, leaving the interpretation of sound duration largely to oral transmission and the individual musician's taste. In a contemporary context, this notational bias, described as "notational perspective" (Bhagwati 2013b), has a more granular scope that can help identify individual improvisation practices. Networked notation technology further extends this concept by allowing the creation of dynamic notational perspectives that can be modified in real-time.

Conventionally, a music score consists of notation which acts as a set of instructions for performers on how to realise a piece of music. Networked notation systems introduce a new paradigm, enabling heterogeneous clients to connect and exchange data and events over a computer network. In such systems a music score can be modelled as a collection of data and algorithms driven by scheduled or triggered events. This dynamic approach allows for interactive and real-time music-making.

In networked music performances, all participants, including musicians, composers, conductors, audience members, and digital audio and video engines, can communicate with each other through the network in real-time. This democratisation of information flow blurs boundaries between traditional music-making roles as any participant in a performance can be assigned decision making or sound production agency.

In this networked environment, music-making takes on a communal and collaborative nature. Audience members become active participants by engaging in the performance through their personal mobile devices. Composers and musicians can interact in real-time to modify the composition flow. Conductors can adjust generative algorithm parameters to alter the notation sent to performers. A computer running an algorithmic engine can trigger score visualisations on audience mobile devices, enhancing the overall immersive experience of the performance. This interconnectedness and fluidity of roles provides a unique and interactive music-making experience reminiscent of ancient communal music-making practices.

A heterogeneous client environment requires multiple score representations as each network client might require a different input type. For example, a score on the musicians' devices might be rendered as a symbolic notation whilst the same score on the audience's devices might be visualised as an animated graphics.

Conditional Love (2016) by Simon Katan is an audiovisual work influenced by the gaming participation paradigm. It involves a laptop performer sitting in the designated stage area in front of a large screen and an audience equipped with personal mobile devices. The audience's devices

are connected to the main stage computer via a local area network, allowing Katan to control the musical and textual narrative of the performance by playing soundtracks, sending instructions to the audience, and initiating new game levels. At certain points during a performance the smartphone players produce a significant portion of the music output by interacting with abstract amoeba-like creatures on their devices. Individual creatures eventually become visible on the large stage screen. The players can then navigate their personal avatar across the screen and create their individual trajectory, providing a shared visual experience. Whilst the onsets of the game stages are controlled by Katan, there is no strict time synchronisation between individual sound source outputs or movements in this piece.

In her analysis (2018), Lüneburg concludes that *Conditional Love* has:

- Highly accessible and easy to use interfaces that are counterbalanced by a crude system on the performance side.
- A high percentage of determinism, which makes the system predictable for the performers.
- High system agency which corresponds with the notion of having only little creative leeway.
- Middle of the range ‘liveness’ (how the system responds to actions by the performer).
- More ludic than paidic play, game-driven play is on the foreground of the player’s experience.⁴

Anders Lind has produced a number of compositions for mixed ensembles that incorporate mobile phones as an extension of the orchestra, akin to a digital choir (Lind 2020a). Lind’s mobile phone application displays simple circular or rectangular graphics, split into differently coloured areas that resemble hardware controller buttons. Clicking on a coloured area triggers a mapped sound sample rendered by the mobile phone hardware. The score for the mobile phone choir is displayed on a large screen with graphical shapes identical to those seen by performers on their mobile phones. Animated instructions on the large score indicate to mobile phone choir members what colour to play and when. The orchestral parts for acoustic instruments are printed in a traditional static form. However, conducting information is automated on the large screen, displaying what should be played and when. The score requires rehearsal time for the mobile phone choir to become familiar with the application and performance environment (Lind 2020b).

Dynamic scores with linear stave notation typically either use the full page or stave update. In Richard Hoadley’s *Calder’s Violin* (2011), the entire view is replaced with new notation at once, whilst in Cat Hope’s *Longing* (2011) and Luciano Azzigotti’s *Spam* (2009), the notation moves continually from left to right in a continuous scroll. These strategies are suitable for particular score types. However, for complex notation scores, the full page refresh strategy does not provide much preparation time for musicians, especially when performance continuity is required at fast tempos. The continuous scroll strategy requires musicians to focus on a fixed

⁴The term “paidic” refers to free, spontaneous, and undirected playfulness, whilst “ludic” play involves predefined rules, goals and structure.

point on the screen where the notation crosses a vertical synchronisation line, thus reducing their capacity to look ahead and prepare for upcoming changes. It also requires continuous notation availability so it is not ideal for generative or free-timing scores.

A networked music score may incorporate data intended for a digital video engine, which usually needs to be translated into a video device-dependent data protocol. Quintet.net Viewer employs Jitter (2003) matrix processing objects and various video processing algorithms that can be dynamically controlled over a network. This allows musicians to control visualisations during a performance.

It is evident that networked music-making expands traditional notational perspectives and introduces new creative possibilities. However, it also increases the complexity of the composition process, necessitating a versatile skill set capable of addressing technical and graphical design challenges.

1.8 Gaps And Research Aims

To address the research questions outlined in Chapter 1, it was first necessary to define the solution's technical and creative scope and develop a methodological approach for its realisation (1.8.1). This framework, presented below, draws on Small's ideas (1998) that question the fundamentals of music-making relationships. Furthermore, it defines the possible modes of participation and outlines the principles applied for both the user interface and action model design. The system design is also informed by the gaps identified in existing solutions (1.8.2). Whenever possible, existing solutions were either reused or enhanced. New software development was undertaken where no suitable existing option was available. Finally, a set of creative and technical objectives that serve as a roadmap for ZScore's development are outlined in the chapter's conclusion (1.8.3).

1.8.1 Solution Framework

For Christopher Small (1998), musicking is about relationships between sounds and people who are taking part, not so much about relationships which actually exist but more about those that we "desire to exist and long to experience". A music performance reflects and shapes these ideal relationships, or values, and allows those who take part to try them and test whether they fit, thus serving as an instrument of exploration. Small poses three questions (1998, p. 193) that define the relationships and, therefore, the meaning of a musical performance as:

1. What are the relationships between those taking part and the physical setting?
2. What are the relationships among those taking part?
3. What are the relationships between the sounds that are being made?

There are various approaches to addressing these questions in the context of networked music-making. The following answers represent an attempt to find solutions to the research questions

stated at the outset of my work (Chapter 1).

1.8.1.1 Relationships between those taking part and the physical setting

One of the aims of this research is to have all music performance participants connected over a computer network. Each participant connects to the network via a physical interface (mobile device, laptop etc.) thereby altering the nature of the relationship between the performance participants and the physical setting, as shown in Figure 1.17. The participant type in Figure 1.17 is indicated by a single letter: A stands for audience member, M for musician and C for composer/conductor.

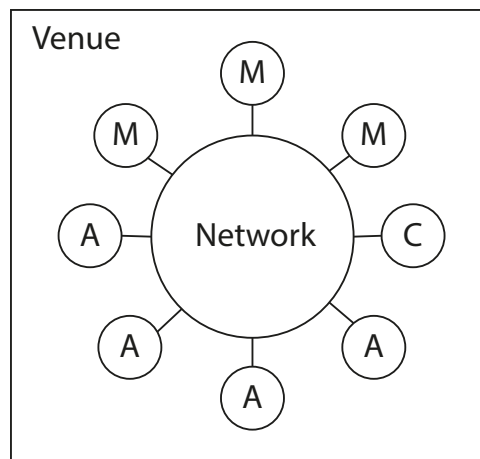


Figure 1.17: Relationships between participants and physical setting

The communication in this setting is multimodal. Apart from direct visual and aural communication, participants also send and receive digital information through their network interfaces. As illustrated in Figure 1.17, the aim is to provide participation and decision-making agency to all connections, regardless of the participant type. The seating arrangements should ideally avoid any distinction between participants, such as a physical separation between the stage and the auditorium. In small scale performances, all performers could sit or stand in a circle as illustrated in Figure 1.17, providing a direct line of sight between all participants. Additionally, all participants would get a broader understanding of the performance context if they could observe how other participant types utilise their score representations.

1.8.1.2 Relationships among those taking part

In Western classical music tradition, the composition process is asynchronous to the performance, and the composer-performer relationship is usually indirect and mediated through a score. Musicians might be aware of the composer's body of work, historical context, and other interpretations of the score, which deepens their understanding of the relationship. However, there is no direct feedback loop between a composer, who might have lived centuries ago, and performers during a performance.

The audience and musicians, on the other hand, establish a direct and synchronous relationship during a performance. All participants react to the aural and gestural performance outputs, creating a shared experience in space and time. However, in a classical music concert hall, participant roles are typically strictly separated. Performers are solely responsible for interpreting the score, whilst the audience is expected to quietly observe and internalise the performance until the conclusion of the score interpretation. In the classical music performance scenario, the majority of audience members have pre-defined expectations of what the performance should sound and look like, due to their familiarity with the repertoire and concert venue rituals. The aesthetic value of the performance is often determined by the comparative evaluation based on previous experiences. Figure 1.18 illustrates the described typical Western classical music-making relationships between participants. Participant types are represented by the letters C (composer), M (musicians) and A (audience). The performance communication in Figure 1.18 is mediated by sounds and gestures produced by performers.

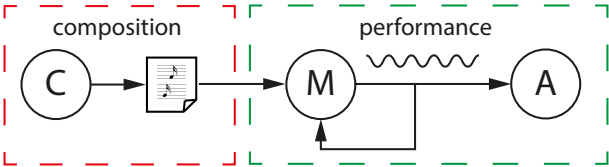


Figure 1.18: Western classical music participant relationships

Music-making in a networked environment opens up a new set of relationships. As with the classical scenario, a composer can create the score asynchronously. However, in a networked performance environment, all participants, including a composer, can interact with the score and other participants in real-time during a performance. Figure 1.19 illustrates relationships between networked music-making participants and the score. Letter D represents any digital engine connected to the networked performance environment. Each networked participant type interacts with the score through a proprietary score representation. In Figure 1.19, a score representation is illustrated by the letter R. A participant type-specific score representation is indicated by a subscript equivalent to the participant type letter (e.g. RM for musicians’ score representation).

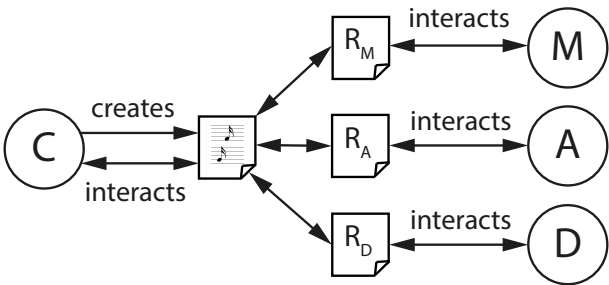


Figure 1.19: Networked score relationships

In addition to the participant-score relationships described above, all participants can create direct communication channels between each other during a performance. A simplified, generic

set of networked music-making relationships is illustrated in Figure 1.20. Specific participant types are abstracted into a common entity marked with the generic letter P, representing any participant type, including a composer. The range of participant types [1, N] is bounded by any positive integer value N.

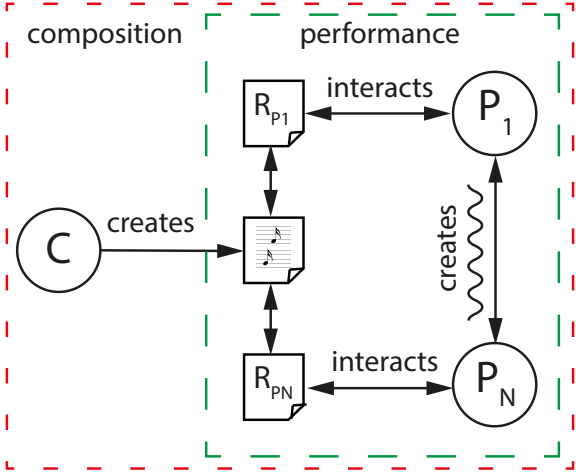


Figure 1.20: Networked music-making participant relationships

As illustrated in Figure 1.20, processes of composition and performance overlap in a networked music-making environment. A composer can create a score fully or partially in isolation pre-performance. However, score content, its representations, or a composition flow can also be created or modified in real-time during a performance by a composer or any other performance participant type. In addition to the communications enabled by the networking technology, performance participants also establish direct aural and visual communication channels during a performance, similar to the classical performance scenario. The difference here, however, is that audience members now become active participants possessing the same participation agency as any other performance participant type. These new features fundamentally change the nature of the relationships amongst those taking part.

1.8.1.3 Relationships between the sounds that are being made

Small’s third question refers to the relationships between basic elements of music. In the extended networked music environment, it is necessary to consider several other elements that constitute a performance in addition to sounds, such as visuals displayed to participants, and dynamic notational or interactive elements. The relationships between these basic performance elements can be fully defined during the composition process or be created during a performance in real-time, depending on the intended improvisation dial position (Figure 1.3).

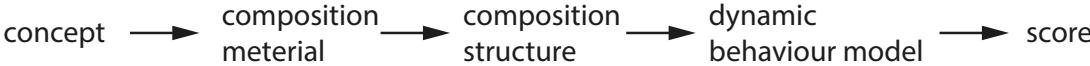


Figure 1.21: Composition process for networked scores

The broad outline of the networked composition process used during this research is shown in Figure 1.21. The process of composition typically starts with the identification of a specific concept, which may be external to the musical context. Composition material and structure are then derived from this concept through a variety of approaches, including objective numerical analysis, parameter mapping, and subjective responses informed by learned behaviours and lived experiences. The order of material and structure creation is flexible, depending on whether a hierarchical, top-down compositional approach is adopted.

In the general case, networked composition material can consist of musical, visual, interactive elements, and algorithms, as presented in Figure 1.22. The letter “M” in Figure 1.22 represents the composition material. Music material includes pitch sets, rhythmical patterns, timbral choices, vertical textures, playing techniques, etc. Visual material consists of any gestures, graphics, or videos displayed to participants during a performance, including various score representations. Interactive material contains all features related to participants’ interactions that result in modification of performance elements in real-time, such as action triggers, their outcomes, dynamic notation overlays, etc. Algorithms incorporate both client and server-side score-related logic, along with various score scripts.

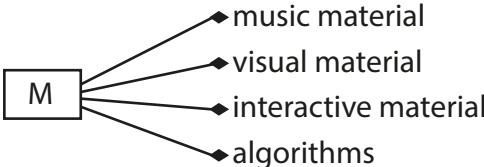


Figure 1.22: Networked scores composition material

The seed composition material (M) is typically defined through the conceptual space mapping process. The seed material is then modified by applying transformation (T) to obtain a new material version (M’). Composition material can also be derived by merging features of two or more different source materials. This process can be repeated a number of times to create the required number of material versions as illustrated in Figure 1.23.

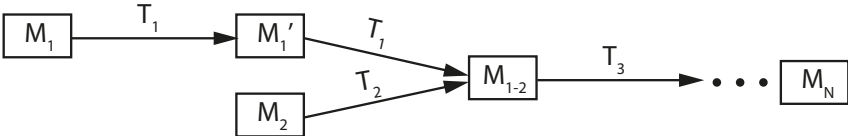


Figure 1.23: Composition material generation process

The composition structure in time is broken down into a hierarchy of time-dependent components, starting with sections that contain a number of pages, which can consist of one or more bars, as shown in Figure 1.24. A page in this context represents the notation view that is active and visible to musicians at a particular moment in time. Bars contain elements that need to be scheduled in time such as notation, scripts, graphics, audio, etc.

The vertical composition structure consists of material elements scheduled to happen at the

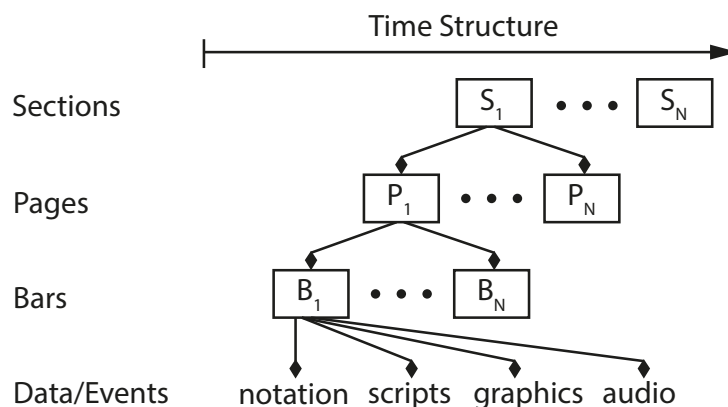


Figure 1.24: Composition time structure components

same time. The vertical structure could contain sounds, gestures, graphics, scripts, and other composition material. Figure 1.25 illustrates the vertical hierarchy starting from the instrumentation consisting of acoustic and digital instruments, as well as various audience mobile devices. Each instrument can be assigned a number of dynamic performance parameters that can be modified in real-time, thereby altering the instrument's sound output. For acoustic instruments, this could include various dynamic notation overlays aiming to modify the instrument's dynamics, pitch, timbre, or playing technique. Digital instruments and mobile devices have similar dynamic parameters, which may include gain, filter frequency or quality factor, effect modification, preset number change, and more.

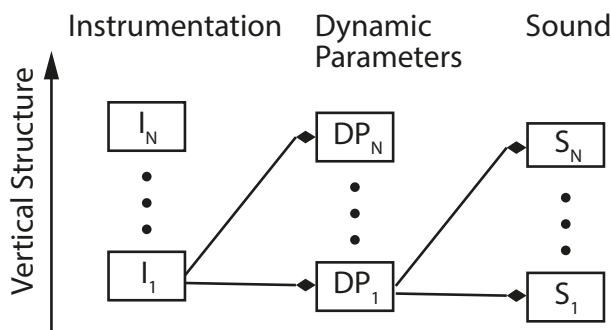


Figure 1.25: Composition vertical structure

The relationships between the composition material groups can be fully defined during the composition process or be created during a performance by the participants or algorithms based on the composition's dynamic behaviour model. The dynamic behaviour model defines who can do what and at what time, effectively managing composition material assignments in the two-dimensional grid illustrated in Figure 1.26. In this grid, the horizontal and vertical axes represent the time structure and the vertical sound structure hierarchy respectively, as explained in Figure 1.24 and Figure 1.25.

Rules defining the order of the composition material play may follow traditional music forms such as binary, ternary, rondo, etc., depending on the compositional intentions for each score. In general, a literal repetition or a modified recapitulation of the material is used as a compositional

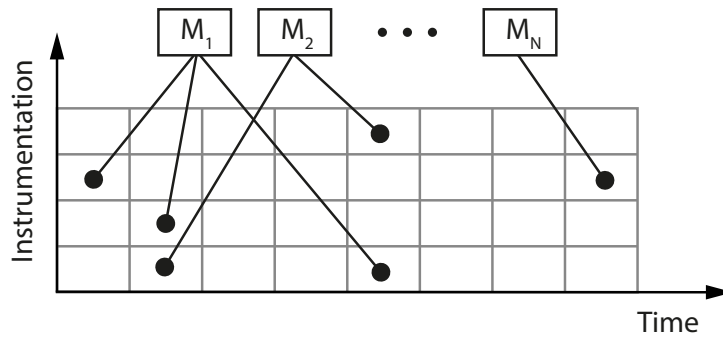


Figure 1.26: Music material dynamic assignment

device in order to provide familiarity when multiple visual and interactive features are utilised in the score.

All the elements explained above that capture music-making and material relationships are a constituent part of a networked composition score. The score, therefore, becomes much more than just a collection of music notes.

1.8.1.4 Modes of participation

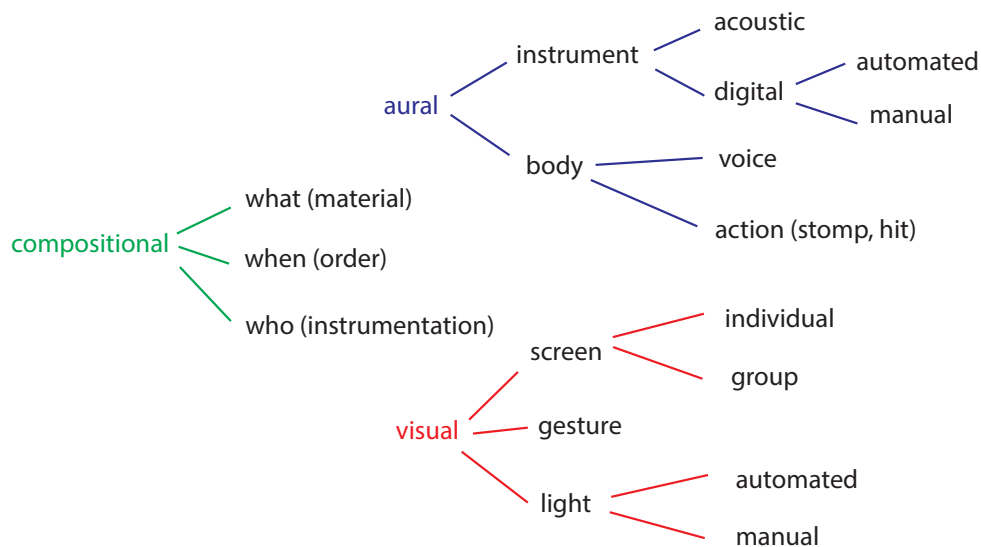


Figure 1.27: Modes of participation in a networked performance

The composition material should utilise the range of possibilities that a networked system can offer. This set of possibilities can be expressed through various modes of participation, as illustrated in Figure 1.27. The illustrated tree is just a starting point that can be expanded with novel participation approaches in the future. It contains three root modes of participation: compositional, aural, and visual. The compositional mode includes any participation that impacts the performance flow, whilst the aural and visual modes relate to the means of sound and visual production respectively.

Any participant type (musicians, audience, etc.) can be assigned one or more modes of

participation in a composition. For example, audience members can be assigned the agency to decide what material should be played next (compositional mode) and use their mobile device as digital instruments (aural mode). Their digital instruments (mobile devices) can be played automatically by an algorithm or manually by an audience member. Musicians (one or more) can be given the ability to choose instrumentation (compositional mode) or trigger an animation on the audience’s mobile devices (visual mode). The intended modes of participation directly impact the nature and content of the compositional material.

1.8.1.5 User Interface Design

In a networked music-making environment, participants view score representation and interact with the system through digital user interfaces (UI). These interfaces typically run on tablets, mobile phones, or laptop computers. The quality and intuitiveness of the provided UI strongly impacts the user experience of the entire performance. McKay (2018) argues that an intuitive UI should be self-explanatory, proposing eight steps that lead towards a well designed intuitive UI, as illustrated in Figure 1.28.

Step	Description	Do	Avoid
Discoverability	Able to find features when needed	Make starting point obvious	Poor layout, too many options
Affordance	Suggests how to perform the action, eg. button	Visual metaphors, consistency	Unnecessary real-world decorations
Comprehensibility	Meaning and effects of elements are understood	Simple language, explicit terms	Jargon, need for memorisation
Responsiveness	Clear, immediate indication of state and results	Show success or failure of action	Colour confusion
Predictability	Foresee results before actions, meet expectations	Consistency with user mental models	Confusion due to surprising results
Efficiency	No unnecessary interaction or repetition	Clustering based on usage, defaults	Poor layout and sizing incorrect defaults
Forgiveness	Preventing mistakes, easy recovery	Reduce impact of mistakes	Focusing only on 'happy path'
Explorability	Use without fear of getting lost or making mistakes	Confirm destructive actions	Unclear navigation model

Figure 1.28: Intuitive UI design principles

A networked music-making UI may potentially offer enough information and functionality to address all the steps listed in Figure 1.28. However, similarly to any relatively new and unfamiliar environment, the predictability of actions with delayed, subtle, or intricate outcomes might not be achievable without comprehensive instructions and labelling. Depending on the score’s aesthetic, the predictability of action outcomes might not be a desirable trait after all. If the UI cannot achieve full intuitiveness for all users, McKay suggests that it should at least maintain consistency and usability in terms of outcomes — such as being comprehensible (users

understand it after trying it once), learnable (requiring experimentation), and guessable (trial and error).

Users interacting with a system for the first time need to construct a mental map of the system's possibilities and the anticipated outcomes of their actions. The user interface functions as a window, providing a view into the entire networked system. From the perspective of interactive system design, however, this is just an endpoint. Any functionality happening behind this view requires further modelling and implementation, both on the server-side and any affected client nodes. The system behaviour resulting from user actions in networked music performance systems should be carefully constructed to facilitate necessary participant interactions whilst also assisting users in forming an internal model of their role in the performance.

1.8.1.6 Action Modelling

An interaction with a musical instrument typically yields an immediate outcome in the form of sound. Similarly, user actions in computer gaming produce instant visual, aural, or haptic feedback generated by the system. These learned behaviours instil similar user expectations when interacting with networked music performance systems. Therefore, the system should provide sufficient preparatory information and appropriate feedback for actions that do not result in an instant or obvious outcome, aiding participants in developing a suitable mental model of action outcome expectations.

Figure 1.29 illustrates a desired Unified Modelling Language (UML) (1997) sequence diagram for user actions resulting in an instantaneous outcome in a networked music performance system. The system components in blue boxes represent server-side components (Score Processor and Scheduler, further described in Chapter 2), whilst the yellow rectangles represent various clients, such as musicians' notation software, audiences' score views on mobile devices, or various audio and video engines. The action destination can theoretically be a server component, however, in ZScore, the destination range is usually limited to clients that produce perceptible outcomes. Closed triangular arrows symbolise synchronous calls, whereas open-arm arrows indicate asynchronous execution.

In Figure 1.29, the call chain begins with the Scheduler component notifying the Score Processor about all actions that can be executed at a specific moment in time, based on the score configuration. The server's Score Processor then dispatches a request to all interested clients to display available score actions to users. A performance participant can then choose to trigger one of the actions presented in their score view. The participant's interaction with their score representation leads to the submission of the start event for the selected action to the server.

In this scenario, the server promptly executes the action by transmitting the execution request to all destinations configured to produce the required outcomes. Depending on the score, these outcomes might involve generating aural or visual signals, music notation, instructions to the audience, or other events. Following the action's execution, the destination clients report back to the server with a confirmation of the action's outcome. This confirmation can be

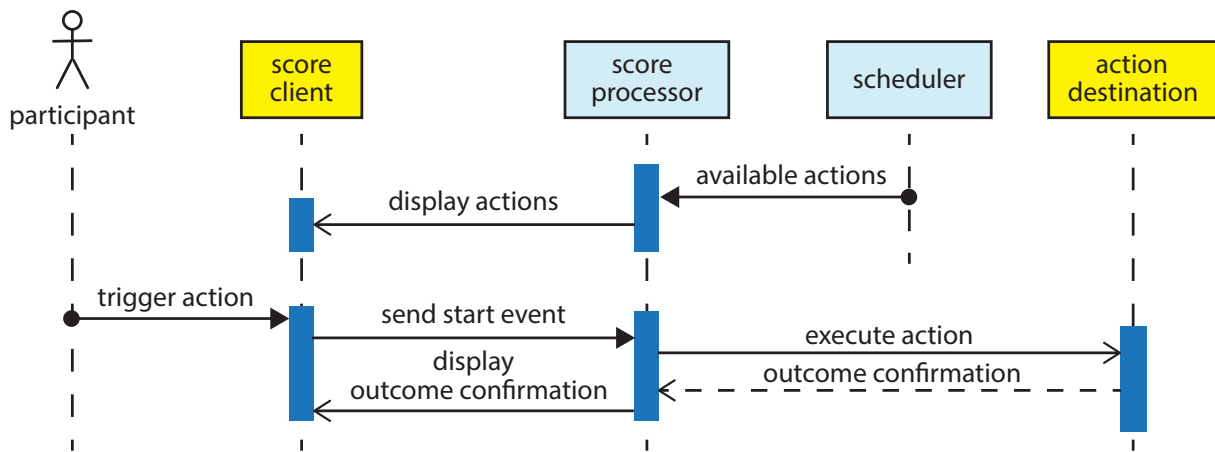


Figure 1.29: Sequence diagram for instant action outcome

either a positive or negative acknowledgment, depending on whether the desired outcome was successfully generated or not. Subsequently, the server conveys the action outcome confirmation to all necessary participants, including the originator of the action.

Principles of intuitive user interface (UI) design require that all user actions should receive prompt acknowledgment in a comprehensible format. Appropriate feedback establishes connections between the actions and the user’s mental model of their interactions with the system. This instils confidence and enhances user satisfaction with the system’s interactivity, and, by extension, with the overall performance.

Any action that leads to outcomes potentially beyond the participant’s mental model necessitates a more comprehensive preparation and sequencing model. Figure 1.30 illustrates the UML sequence diagram for participant actions resulting in delayed outcomes.

The call chain starts again with the Scheduler notification of available actions at a particular point in time. This time, in addition to available actions, the score client displays an outline of outcome expectations. This outline can be represented in a textual format, by simply explaining what the outcomes may be, or in a graphical format where the visualised icon can be inextricably linked to the outcome (e.g., the music note icon resulting in the sound production). Alternatively, the outcome expectations can be conveyed aurally (e.g., before the performance).

The action’s start event is initiated by the participants, similarly to the previous example. However, in this case, the server does not carry out the action immediately. Instead, it schedules the end event to occur at a predetermined time in the future. An acknowledgment of the start event should be presented in a suitable format on all relevant clients.

After confirming the start event, the system should provide sufficient information to all interested participants regarding the current state of any delayed or extended actions. Depending on the score aesthetics, this could involve a straightforward textual indication or a more intricate animated graphical representation of the time remaining until action completion. In Figure 1.30, the visualisation of action state progress on participants’ score views is illustrated within the loop box. The aim of this visualisation is to establish a link between the participants’ physical

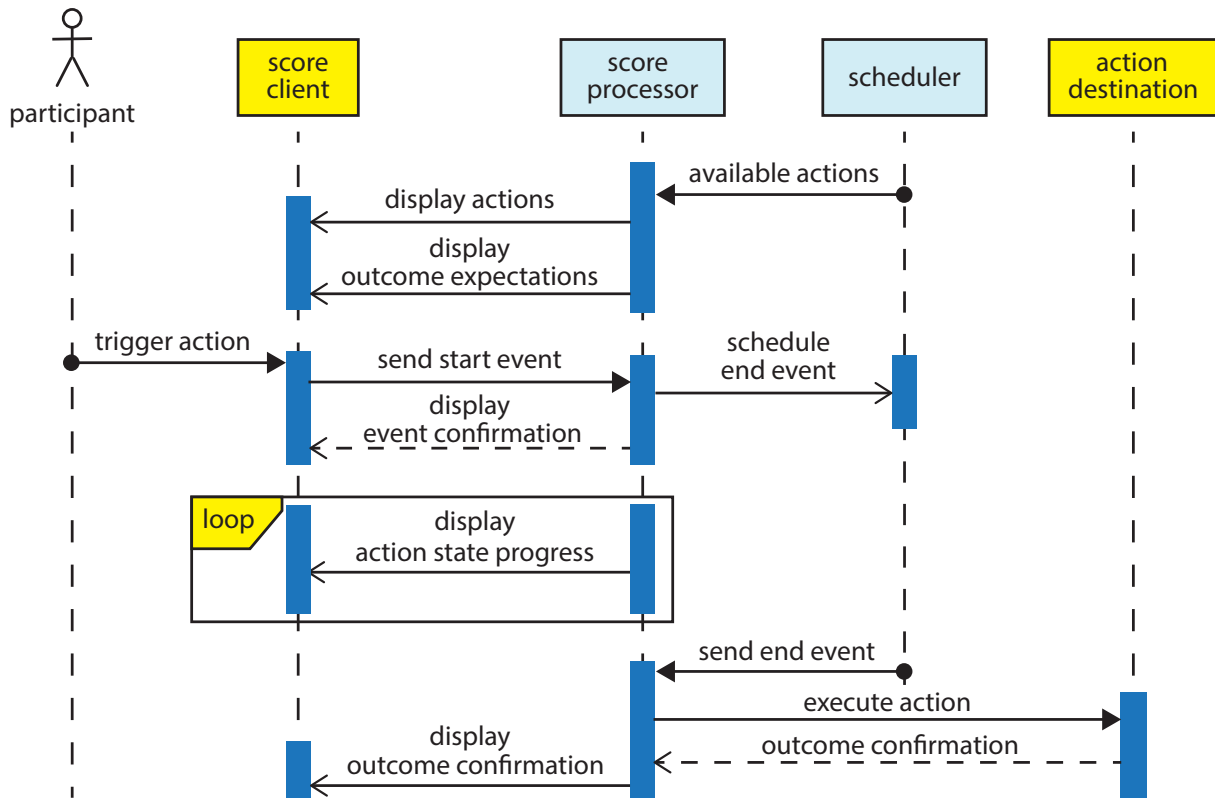


Figure 1.30: Sequence diagram for delayed action outcome

interactions with the system and their anticipation of the associated outcome.

Upon reaching the scheduled end event time, the Scheduler notifies the Score Processor, which finally executes the configured action outcome. Similarly to the previous example, the confirmation of the outcome is displayed on the score views of all relevant participants. The outlined action models have been designed for the ZScore networked notation system, but can be implemented in any system requiring similar interaction behaviours.

Both the UI design and action model should always be driven by compositional intentions, as illustrated in Figure 1.31. The UI design and action model need to work closely together to provide functionality derived from compositional intentions. The interface and mechanism for data and action transfer between the client-side and server-side should be reusable, however, the UI look and feel, its functionality, and interaction modes may vary significantly between different compositions, particularly when related to audience participation.

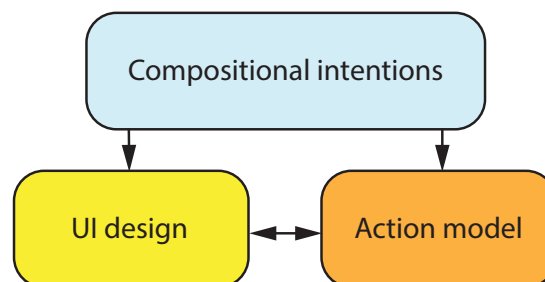


Figure 1.31: Compositional intentions, UI design, and action model relationships

1.8.2 Gaps

Research into existing networked music composition and performance solutions revealed several gaps in the required behaviour. Most of the existing solutions provide a one-way communication flow (Figure 1.2). Whilst there are some interactive features in existing solutions that allow players to send feedback to performance controllers, there is a lack of a comprehensive interactive music composition and performance model for multi-directional communication flows (Figure 1.12) that includes event distribution rules such as: who can take action, at what time, what the action outcomes are, and who the outcome recipients are. An interactive model needs to accommodate different event distribution modes, such as one-to-many, many-to-one, or many-to-many. This requires precise and flexible event scheduling and a rule definition mechanism. The implementation of an intuitive UI (Figure 1.28) and participant action models (Figures 1.29 and 1.30) would also necessitate further development in existing notation solutions.

The premise that any participant can assume decision-making agency blurs boundaries between traditional music-making roles. An interactive music composition model needs to cater for multiple role definitions and their mapping to music material, available actions, action timings, and outcomes. The impact of non-linear composition flows and a multitude of action outcomes on compositional aesthetics needs to be carefully considered. Interactive features (buttons, controls, etc.) need to be clearly visible and easily accessible on musicians' and audiences' front ends. Users need time to notice and react to interactive features and, ideally, understand the intended impact of their actions.

A dynamic notation view requires a carefully thought-out refresh strategy that does not interfere with the currently played notation, providing enough time and space for musicians to prepare for the upcoming material. The refresh strategy needs to take into account network and rendering latency, ensuring that notation updates do not disrupt the score continuity.

A good dynamic score front end design should fully utilise the available screen real estate and provide a clear view of the notation, available actions, and any additional information that musicians should be aware of during a performance. If delivery to heterogeneous platforms is required, the notation should remain legible when scaled to any of the common screen aspect ratios (4:3, 16:9 and 16:10), screen sizes (10 to 17in) and resolutions (1024x768 to 3840x2160).

An unconstrained mixture of complex symbolic and graphic notation with precise synchronisation is challenging to achieve with current networked music tools. Scores are either entirely graphic or mostly symbolic with optional overlaid graphics. The composition process is usually completed offline prior to a performance, without a provision for real-time modifications during a performance. Real-time interactive control of performance parameters or compositional structure is difficult to achieve.

Musicians' notation devices are most frequently connected to a network over Wi-Fi. Whilst this is a simple and convenient solution, it is only suitable for scores where latency spikes of more than 10 milliseconds do not have an impact on the composition aesthetics. Wi-Fi intermittent

latency spikes due to interference and internal message queuing can occasionally reach hundreds of milliseconds. Similarly, any system relying on NTP clock synchronisation over the public Internet, or a local network with significant congestion, can produce similar delays.

The score state in existing solutions is typically held on the client-side. For example, Drawsocket (Gottfried and Hajdu 2019) and Decibel ScorePlayer (Hope et al. 2015) notation clients load necessary score data prior to the performance into local memory. Clients then receive performance management events, such as start or stop, over a network. Any score state change has to be managed on the client-side separately for each notation view. This design makes it very difficult to manage the score state in a scenario where any network node can send events that impact the state of any other node. In this case, a server-side score state management implementation is more beneficial. With this approach, all real-time changes happen in one place and all clients are updated when necessary. Other advantages include reduced network congestion due to the smaller amount of data held on the client-side, and easier reconnection and synchronisation of client views during a performance.

Interactive music scores managed on the server-side need to provide simultaneous access to shared data for multiple users. In order to protect shared data integrity, a common engineering approach is to allow only one user thread at a time to access shared resources and block all others until the active user thread's processing is completed. Unfortunately, this approach introduces additional latency and can have a significant impact on the performance flow. Several non-blocking algorithms exist in industries where high-frequency low-latency solutions have been utilised for many decades, such as the LMAX Disruptor pattern (Thompson et al. 2011). The implementation of non-blocking data access is a requirement for any scalable networked performance system.

Intermittent message loss in networked system communications can occur for various reasons, especially when a non-guaranteed UDP protocol is used for message dissemination. An interactive networked music-making system should have a built-in strategy for dealing with message loss to ensure an uninterrupted performance flow. Several approaches can be taken, including reliable UDP Multicast (PGM, TRDP, LBT-RM, etc.), TCP/IP point-to-point connectivity such as WebSocket, application message redundancy (replay of missing data and events), and other similar solutions.

1.8.3 Objectives

The project aims to achieve the following creative objectives, in line with the outlined research questions:

- Create an innovative compositional approach that leverages acquired knowledge and lived experiences by utilising state-of-the-art networking technology as a creative extension of human communication capabilities in music-making.
- Pursue a concept-driven composition method, utilising creative isomorphic mapping be-

tween conceptual and compositional structures and materials.

- Reflect on the relationship between personal states of consciousness and contemporary society, including contradictions between subjective and objective values.
- Create a musical relationship triangle with musicians and the audience, and develop a compositional approach to explore it.
- Expand the performance participation paradigm, democratising music-making by allowing all participant types (audience, musicians, conductor, etc.) to assume decision-making or sound production agency.
- Enable musicians to fully and freely express their own learned experiences and knowledge within a compositional context, allowing them to bring different skill sets to a performance.
- Explore the full range of the decision-making spectrum, from fully composed to freely improvised, whilst retaining compositional identity.
- Merge acoustic and digital sound sources into a coherent sound field whilst preserving the sonic qualities of individual sources.
- Develop engaging interactive score representations for all performance participants that convey compositional and artistic aims.
- Implement different aspects of audience participation in a performance, such as real-time decision-making, sound production, physical gestures, etc.
- Continue the development of mixed sound-action notation by providing dynamic control of individual performance parameters for each instrument.
- Explore the full range of unstable non-tempered sound textures created by the separation of performance parameters and their dynamic control.
- Develop and explore a dual “composer - performer” role in a networked performance environment.

The ZScore networked music-making system, specifically developed for this research as a platform for delivering the artistic intentions listed above, ultimately aims to offer:

- An interactive networked music composition and performance environment providing connectivity, participation, and decision-making agency for all participants.
- System support for scores that allow participants to assume different roles and impact a performance in real-time.
- Complex symbolic, graphic, or mixed notation authoring for any instrumentation (e.g., full orchestra) and type (acoustic, digital, algorithmic, etc.).
- Dynamic, intuitive, and interactive networked notation views on heterogeneous clients for multiple participant types, allowing automated notation updates, animation, position tracking, event triggering, clear notation visibility, user preparation time, and real-time improvised music-making.
- The ability to control a dial on the decision-making spectrum (Figure 1.3) in real-time.
- Reliable and scalable low-latency messaging where critical events are delivered and exe-

cuted with humanly imperceptible latency (sub 10 ms compound network and application latency).

- Accurate performance synchronisation across all networked nodes, which includes the effects of network latency and jitter.
- Efficient score data encoding and segmentation strategies that minimise transcoding and avoid packet loss during network transport.

Chapter 2

ZScore

ZScore is a networked music composition and performance system developed to achieve the objectives outlined in Chapter 1.8. The system architecture, along with its key components and features, is further described below. The system comprises both third-party and newly-developed components that collectively create an interactive environment, enabling the performance of complex dynamic scores and providing connectivity and decision-making agency for all participants in the performance.

The proprietary components developed specifically for this project include the ZScore server, Control GUI, and various client-side libraries: the Adobe Illustrator JavaScript plugin, Max mxj external and patches, as well as various JavaScript libraries for INScore and web browsers. The ZScore server architecture is further explained in this chapter, whilst the other components are introduced gradually in the composition commentaries.

Figure 2.1 depicts a typical ZScore performance system layout for venues with up to 100 audience members. Depending on the score, additional components such as video engines or motion tracking devices can be added to the system. For larger numbers of users, additional hardware components may be required, as further elaborated in Appendix A.

The local area network (LAN) on which the performance system runs is divided into two virtual local area networks (VLANs): internal and external. VLANs effectively isolate network traffic, providing the illusion of separate networks whilst utilising their shared underlying physical infrastructure. This separation enables enhanced network management, security, and efficiency. The network traffic from the external VLAN is filtered by the firewall, removing unnecessary or potentially dangerous messages.

All latency-sensitive nodes, including the ZScore server, audio engines, and musician tablets, are connected to the network through Ethernet cables (the green connections in Figure 2.1). This approach ensures the fastest and most reliable data delivery for notation rendering, audio management, and decision-making by musicians and conductors during a performance. Audience mobile devices connect via Wi-Fi, and therefore, the audience score representation has to take into account potential delays in network message transmission caused by interference. However, Wi-Fi provides the most versatile solution for audience member connectivity due to its

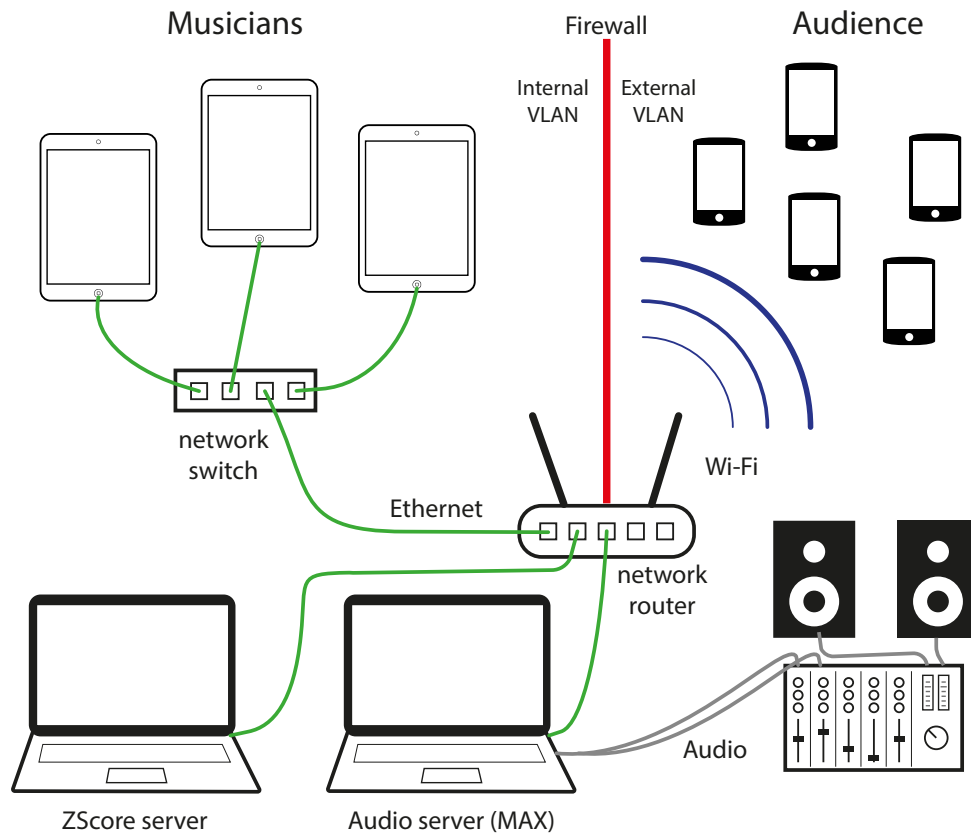


Figure 2.1: ZScore performance system components

widespread availability on mobile devices.

All HTML5 clients, including the musician’s notation and audience’s score views, were newly developed specifically for the ZScore’s composition model. The notation view utilises an innovative alternating pane layout, further described in Chapter 3.2.1.2, and provides interactive features explained in Chapters 3.5.1.1 and 3.6.2.1. A set of reusable JavaScript libraries, described in Chapter 3.5.1.2, provide functionality required for audience score representations. Score-specific logic was further developed in JavaScript for each score that required additional functionality.

In several portfolio pieces, the ZScore audio engine hosted in Max sends out a stereo audio signal that is amplified and delivered through speakers positioned between the musicians, effectively acting as another sound source on the stage. A subwoofer is typically deployed for more accurate lower frequency reproduction. As a deliberate artistic choice, acoustic instruments are not amplified in order to maintain the direct link between a sound and its source. A proprietary Max mxj object, described in Chapter 3.5.1.3, enables network communications, making ZScore Max patches suitable for both event-driven and manual sound generation.

Figure 2.2 illustrates the high-level functional components of the ZScore software. All server-side components are implemented in Java programming language (Gosling and McGilton 1996) and utilise techniques associated with high-throughput, low-latency financial trading systems. The core components – Score Processor, Scheduler, and Transport – handle all internal

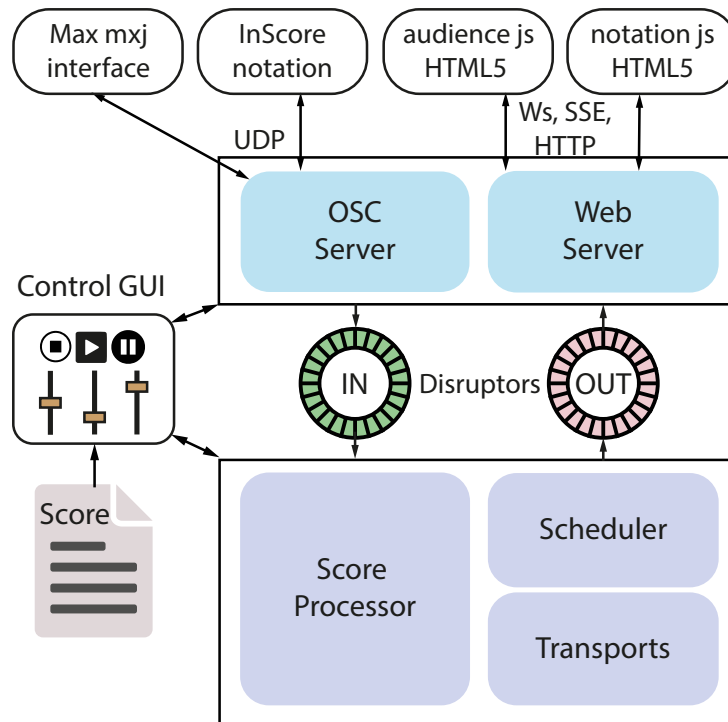


Figure 2.2: ZScore high-level software component architecture

processing related to score data and the timing of events. The server architecture, all the core components, and the interactive composition model have been developed specifically for this project. The core single threaded processing is separated from multi-threaded in/out handlers by two Disruptor pattern ring buffers. This provides the guaranteed ordering of incoming and outgoing events based on their timestamp, as well as fast block-free processing between the core components with minimal memory footprint.

In a simplified form, the Disruptor pattern architecture consists of three components illustrated in Figure 2.3: input Disruptor, output Disruptor, and composition logic, which includes all the server-side core components mentioned above. It is important to note that the composition logic runs on a single thread, as illustrated by the squiggly arrow in Figure 2.3. ZScore maintains score state on the server-side within the Score Processor, which constantly receives and distributes updates that may modify the score state. Single thread processing ensures optimal performance by avoiding score resource contention and data access blocking.

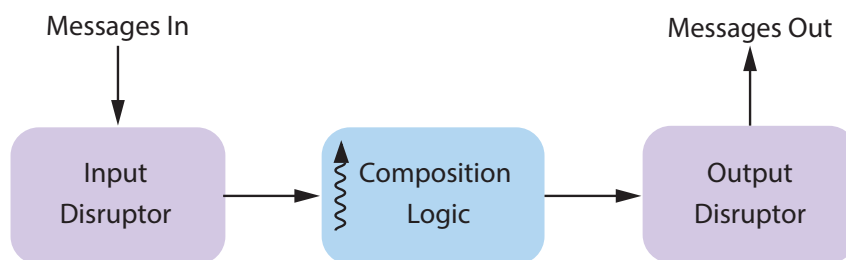


Figure 2.3: Disruptor processing structure

Instead of standard input and output blocking queues used to pass data between threads,

Disruptor uses a single data structure – a ring buffer. It is implemented as a fixed size array that behaves as if it had a circular shape, in which the last element in the array is connected to the first element.

Each ring buffer can have one or more producers and consumers. Producers write data into a ring buffer and consumers read that data. Figure 2.4 illustrates a ring buffer with one producer (Receiver) and multiple consumers (Journaler, Converter, and Composition logic consumer). All producers and consumers run in parallel on a dedicated thread. Each producer and consumer has a sequence counter to indicate which slot in the buffer it is currently working on. They can only write their own sequence counter but can read all other sequence counters. This way the producer can read the consumers' counters to ensure the slot it wants to write in is available without any locks on the counters. Similarly a consumer can ensure it only processes messages once another consumer is done with it by watching the counters.

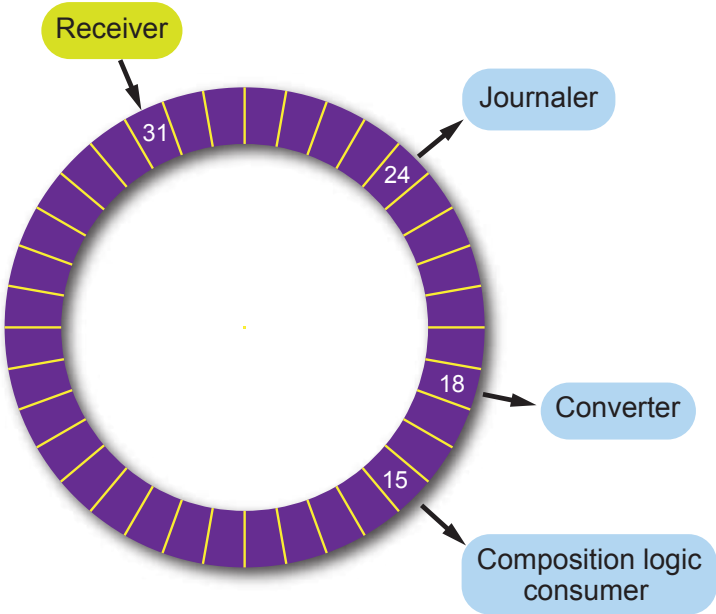


Figure 2.4: Disruptor data access sequencing

In the case illustrated in Figure 2.4, the Receiver is currently writing into slot 31. It then keeps advancing forward, making sure that it doesn't wrap past the composition logic consumer. The Journaler is currently reading slot 24. It stores related message data into an archive for future replay and debugging purposes. The Journaler can read data in a batch up to slot 30 at this point. The Converter takes raw message data received from the network and converts it into a composition logic friendly format in a process called unmarshalling. It can also read up to slot 30 at this point but cannot go past the Journaler. Similarly, the composition logic consumer cannot go past the Converter, Journaler, and Receiver. The composition logic consumer reads data written by the Receiver and sends it to the Score Processor, where most of the score logic is implemented.

Another advantage of the ring buffer is that memory can be pre-allocated by initialising each slot object on startup, thus ensuring a stable memory profile that avoids excessive garbage

collections. The Disruptor provides interfaces for application-specific implementations that map incoming/outgoing data into an internal data object structure.

Internal components, such as the Scheduler, can also produce input events for the Score Processor. As the Scheduler runs in its own thread, in order to avoid thread contention, these events are also submitted via the input Disruptor and executed on the composition logic thread.

The Score Processor consumes input data and produces output events that are sent to clients over the output Disruptor. The output Disruptor functions in a similar way as depicted in Figure 2.4, however, instead of the composition logic consumer, it has output consumers such as Web and OSC servers. The events from the output Disruptor are distributed to all connected network clients such as musicians' tablets, the audience's mobile devices, and Max patch via the Web or OSC servers.

The ZScore Web Server is built on top of the lightweight and versatile JBoss Undertow (2014) implementation. Both the server and miscellaneous JavaScript clients support WebSocket, Server-Sent Events (SSE), and the HTTP polling connectivity mechanism. The connection mechanism preference and fallback order is configurable in the client settings.

WebSocket connectivity is a point-to-point connection providing the most reliable message delivery. However, some firewall configurations may not allow fully-duplex WebSocket communications. Additionally, because WebSocket connections are permanent, the demand for server-side resources increases proportionally with the number of connections. For medium to large size audiences and ensembles SSE is a preferred connectivity solution as it is relatively lightweight and implemented on top of a ubiquitous HTTP protocol and is therefore, by default, widely supported by most firewalls.

The OSC processor utilises the UDP protocol for event distribution, which offers a more scalable communication solution than the TCP/IP protocol as it does not require point-to-point connectivity. However, message delivery is not guaranteed, so it requires strategies to cope with message loss. ZScore provides message replay and frequent time synchronisation across all connected nodes to alleviate related issues.

Core component processing is transport agnostic due to the clear separation of the external connectivity layer. For the purpose of scalability and security, the web servers used by the audience and musicians are logically separated and accessed on different ports. The configuration of these web servers, including the root directories for audience and score content, is stored in the `zscoreApp.properties` file, which is included in the server deployment.

Currently, all server-side components run within the same process. Due to the modular interface design, the external communication layer can be separated into separate processes, including both web servers. This architecture would ensure improved scalability and the uninterrupted operation of the core components in the case of any external denial of service attacks.

The Score Processor is the central component of the performance system, making decisions related to score rules, and determining what data and events should be distributed to whom and at what time. Its implementation follows the delegation pattern, where each score specifies a

delegate in charge of its execution logic. Scores can be configured to either reuse the base delegate implementation or employ a custom delegate depending on the complexity of the required processing. The configuration for score-specific processors is stored in the `zscoreApp.properties` file, alongside other server-side configurations. The Score Processor creates time-dependent tasks based on the score rules and sends them to the Scheduler, where they are dispatched for execution at the required times.

Scheduler and Transport are responsible for all timed operations in the ZScore system. The minimal scheduling resolution is 1 millisecond, which means that task execution can only be delayed by up to 1 millisecond from its intended timing. This scheduling resolution is finer than that of most other similar applications (e.g. INScore's scheduling interval is 10 milliseconds), ensuring that all time-sensitive operations occur with delays imperceptible to humans. However, depending on the size of the task queue and logic complexity, the task processing itself can cause delays. ZScore is configured to report any task processing delays of more than 5 milliseconds for diagnostics purposes.

Transports in ZScore translate relative time in milliseconds into tempo-relative time, such as beats, bars, pages, etc. ZScore supports multiple transports for poly-temporal scores. Events can be scheduled either in elapsed time expressed in milliseconds or in tempo-relative time expressed in Base Beats. ZScore's concept of Base Beat is expressed as a fraction of the whole beat, usually $1/8$. This means that any score event can be scheduled with $1/8$ th (quaver) precision within a score.

This functionality is further extended to provide the tempo-relative time synchronisation across all notation clients. The server sends a synchronisation message to all registered notation clients at configurable time intervals expressed in Base Beats. This approach is similar to MIDI beat clock implementation (Malouin 2020). The synchronisation message contains the current score time expressed as the number of Base Beats since the play started. This timestamp can be used to set the client's position or as a time correction mechanism for local clocks, as described in Chapter 3.2.1.3. If multiple Transports are used, each Transport object sends synchronisation messages at different time intervals, depending on the Transport's current tempo.

The Control GUI is currently implemented as a JavaFx (2008) front end, running in the same process as the server. However, the design allows for its separation into a stand-alone application connecting via a communication layer like all other clients. The Control GUI is responsible for managing various aspects of the score performance, such as loading a score, managing performance participants, starting/stopping play, setting position, changing score parameters, and sending data to other participants.

Complex scores can have a dedicated tab for performance management, including comprov-isation controls that manage the content of dynamic notation regions in real-time. All score management tabs share the same basic transport controls, such as play and stop. This allows for comprehensive control over the performance aspects of the scores, enabling the smooth and efficient coordination of the music-making process.

The ZScore source code and related materials are available in GitHub repositories as outlined in the table below.

GitHub link	Description
https://github.com/szagorac/szcore	ZScore Java server and GUI source code
https://github.com/szagorac/scores	Various materials related to portfolio scores
https://github.com/szagorac/zsweb	Web ZScore and Audience Web view source code
https://github.com/szagorac/zsmax	Max client and patches used in portfolio scores

More detailed explanations of particular aspects of ZScore are available in the Portfolio Chapter. Objectives necessitating additional software development are discussed in the individual score commentaries.

Chapter 3

Portfolio

From A Static Score To Interactive Music-making

Link	Comment
Scores in pdf format	For information only. Most of the scores are visualised on digital devices.
ZScore packages for each score	Download a package in zip format, unzip in any directory, and follow the included User Guide.
GitHub repositories	GitHub root for zscore, zsweb, zsmax, and various score materials.

The portfolio compositions demonstrate a journey that begins with a conventional static score and ends with a dynamic composition model that implements the objectives outlined in Chapter 1.8. Each composition addresses specific challenges that gradually build towards a networked music-making system offering meaningful performative involvement and decision-making agency for all participants.

The journey begins with *Red Mass* (3.1), a conventional static score printed on paper and distributed to musicians before the performance. It serves as an illustration of my approach to concept-driven music composition and aesthetics, which utilises a dialectic between objective and subjective compositional methods. This approach employs an interplay between the constraints imposed by formal structures and subjective responses stemming from learned behaviours and lived experiences. As such, it broadly aligns with the principles of metamodernism, as further explained in Chapter 3.1.

The first composition written for the ZScore system was *Ukodus* (3.2). This development required defining and implementing the entire networked composition and performance process, including score authoring, real-time notation scheduling and distribution, and dynamic notation rendering on computer screens. For notation authoring, I chose Adobe Illustrator software as the most suitable option for generating complex scores in both SVG and raster formats. To

further enhance efficiency, I developed a new Adobe Illustrator JavaScript plugin that facilitates notation authoring workflows and proprietary score data export (3.2.1.4). For real-time event scheduling, distribution, and performance control over a network, I designed and developed the ZScore server and Control GUI using Java programming language (2 and 3.2.1.6). The server provides connectivity for all participants and runs the composition logic. It ensures the ordered processing of messages coming from multiple sources and utilises non-blocking algorithms to achieve high-throughput, low-latency message distribution. After analysing existing networked notation solutions, I selected INScore standalone client for dynamic notation rendering tasks due to its native support for OSC, JavaScript, and a useful time-space mapping implementation (3.2.1.5). The notation is displayed in an innovative alternating pane layout (3.2.1.2), allowing left-to-right, top-down notation reading that is familiar to classically trained musicians, easing the transition from static paper scores. *Ukodus*, successfully performed by Moscow Contemporary Music Ensemble in 2017, served as proof of concept for the ZScore system.

Building upon the system features developed for *Ukodus*, *Vexilla* (3.3) introduced an additional part containing the score visualisation material created specifically for the audience (3.3.1.1). The audience part, embedded within the Adobe Illustrator score, contains scripts that govern audience-specific score visualisation in sync with other music events defined in the score. The visualisation, consisting of various animated SVG elements, is rendered by INScore standalone client in real-time and projected onto a canvas placed on the stage. Additionally, *Vexilla* introduces a system of colour coding in the music notation for specific playing techniques (3.3.2.1). For example, blue lines are used to indicate the timing and pitch of clarinet multiphonics.

Comprov (3.4) explores a novel concept of dynamic notation overlays specifically designed for improvised music-making. To achieve this, I redesigned the instrument stave layout to visually separate performance parameters like bow speed, pressure, and position within a two-dimensional space (3.4.1.1). These areas could then be dynamically overridden by notation overlays controlled by a conductor in real-time (3.4.1.2). This innovative technique allowed for a limited composition material to be looped and drastically altered throughout an extended performance, resulting in a wide range of musical outcomes. To provide required real-time functionality, I enhanced the ZScore Control GUI with dynamic overlay controls (3.4.1.3). Additionally, I improved the ZScore server design to incorporate a generic Strategy interface, enabling system behaviour modifications based on specific score requirements. Two such Strategies were implemented for *Comprov*: Randomisation (3.4.1.4) and Continuous Play (3.4.1.5).

Union Rose (3.5) introduced significant advancements in the participation model, notation, digital audio integration, and technical implementation. For the first time, audience members could connect to the ZScore performance system using their mobile devices, enabling participation in a performance through an audience-specific interactive score representation accessible via any web browser (3.5.1.2). This allowed the audience to participate in decision-making and

sound production either through their devices or actions when prompted by the score. I also implemented a web-based version of the musicians' notation that can run on any digital device for *Union Rose* (3.5.1.1), replacing INScore standalone client. The notation segmentation, initially used in *Comprov*, was further enhanced to provide a cleaner visual layout with interactive features that also allowed musicians to participate in the decision-making process. Furthermore, I developed custom Max objects, including an external object for network communications written in Java, a jsui visual interface for score information, and a set of patches for audio processing (3.5.1.3). ZScore Max patches were designed to be controlled in real-time either by network messages or directly through MIDI controllers. Both the score and server-side processing were enhanced to enable the integration of all of these components into a coherent, synchronised composition model. This included a comprehensive score state management system that allowed play to start from an arbitrary position in the score (3.5.1.5). These advancements inevitably increased the complexity of the composition process, demanding significant technical and creative effort.

Socket Dialogues (3.6) takes a step further towards democratising music-making roles. It consists of a series of musical dialogues designed to allow any number of musicians, regardless of their instrument, to participate. Musicians can choose dialogues, their order, and a role they wish to perform within each one (3.6.2.1). They also retain the flexibility to change their role during the dialogues. Each dialogue role is associated with specific music material. To accommodate this notational flexibility, an innovative "pitch line" instrument-agnostic notation system was invented for *Socket Dialogues* (3.6.2.2). Furthermore, audience members are empowered to provide instant visual and auditory feedback to all participants via their mobile device, leading to continual adjustments to the performance output (3.6.2.5). A unique audience-led musical dialogue (*Interlude* 3.6.3.6) can be performed between any two composed dialogues. Here, the audience acts as a mobile device ensemble, whilst the musicians and conductor react to the audience generated sound through a combination of free improvisation and the interpretation of the dynamic notation generated in real-time. The composer/conductor can also become a performer during *Interludes* by engaging with ZScore Max patches. Despite the inherent role flexibility and variations in notational specificity – with some dialogues requiring precise pitch and timing and others allowing for significant interpretive freedom – *Socket Dialogues* aims to preserve the compositional identity in all performances, regardless of the instrumentation, number of musicians, or audience size.

The following chapters provide more details about each portfolio work. All chapters share a consistent structure, beginning with the composition's instrumentation and a table containing links to relevant downloadable content and recordings. Next, the specific objective for each piece is outlined, including a description of the most important new features introduced in the composition. Each chapter concludes with a score commentary, detailing the composition material and processes applied in each work.

3.1 Red Mass

For:

Flute (C Flute, Bass Flute and Piccolo)
Clarinet (B^b and Bass Clarinet)
Percussion (Marimba and Tibetan Bowl)
Piano
Violin

Duration:	ca. 7 minutes
Links:	Full Score Workshop recording (up to bar 191)

3.1.1 Objective: Compositional Method

The starting point of my journey towards networked music-making, *Red Mass* was written for a workshop with Ensemble Interface in 2015. It is a conventional, static, linear score written in Sibelius software, printed on paper, and manually distributed to musicians and the conductor. Although it does not contain any networked elements, *Red Mass* illustrates my approach to music composition and aesthetics, which has been applied and developed further throughout this research. This overarching approach, applied across all portfolio pieces, is further explained in the following chapter.

3.1.2 Approach To Music Composition And Aesthetics

Contemporary attempts to define the nature of today's post-postmodern world often merge features attributed to modernism and postmodernism. Akker and Vermeulen's *metamodernism* (2010) metaphorically illustrates this as a pendulum constantly oscillating between the seriousness of modernism and the ironic playfulness of postmodernism. Unlike the post-modern ironic detachment of the late 20th century, recent metamodern thought draws from modernist aspirations for fresh forms of expression and innovative solutions, both creative and technical, particularly when dealing with the global challenges confronting humanity. Guido van der Werve's short film "Nummer acht, Everything is going to be alright" (2007) succinctly portrays the current human condition of being compelled to keep moving forwards in the face of global issues.

Critiques of modern society often highlight that the pervasive presence of technology and the constant connectedness facilitated by the Internet, mobile phones, interactive television, and similar media can lead to superficial and shallow participation in culture (Kirby 2015). As outlined below, one of the primary motivations underpinning this research is to advance the integration of technology and enhance the participation experience in the processes of music composition, performance, and listening.

My approach to composition engages a dialogue between objective compositional techniques, such as the numerical analysis of a given concept, or the mathematical modelling of musical parameters, and subjective elements stemming from personal experiences and knowl-

edge. As such, this approach exhibits a broad alignment with the principles of metamodernism, which embrace both irony and sincerity, or the blending of tradition and innovation. It attempts to avoid the inauthenticity of polystylism, where composers may appropriate elements from different musical traditions without fully understanding the underlying cultural context. I draw upon authentic lived experiences and learned behaviours, whilst constantly seeking new forms of expression. This exploration encompasses various elements, such as extended playing techniques, dynamic mixed notation, improvisation, audience participation, the merging of analogue and digital sound sources, real-time audio processing, and others further discussed in this thesis.

One of the aesthetic goals is to treat acoustic and digital sound sources as equals concerning their physical and textural position within the overall sound in a live performance environment. Often, stereo sound amplification in electro-acoustic performances creates an artificially blended sound field that removes a direct connection between the sound producer and the listener. Whilst this might be suitable for some musical styles, my preference is to establish direct relationships between musicians and the audience through aural, visual, and digital communication based on their physical location.

The composition process begins with defining a specific concept, typically external to the musical material, which acts as a seed for the compositional development. Unlike the conceptual art approach, where the concept is in the foreground, the goal here is to create a platform for the development of larger scale musical works by establishing a framework for mapping between the concept and compositional structure and material. This process is further explained in the score commentary for each portfolio composition. For example, in *Ukodus*, a sudoku puzzle is used to construct the entire composition structure, whilst in *Union Rose*, the composition structure is derived from symmetric features of a church window rose. Concept-related ideas can be conveyed visually through participants' score representations, in addition to the musical and gestural information.

The musical mapping of ideas drawn from this defined concept leads to the creation of a top-down compositional structure. Rhythmic, harmonic, textural, and gestural components are then derived from these formal structural constraints and subjective responses to the composition concept. These subjective responses may include personal experiences, learned behaviours, conscious feelings, or subconscious emotions. For example, in *Vexilla*, the pitch and rhythm are derived from songs that I associate with particular flags, such as “Na Planincah“, a Slovenian folk tune and one of the first melodies I learnt to play on the piano, and “Kad ja podjoh na Bembasu“, a song from my hometown that invokes a strong internal emotional response. However, the temporal structure in both cases is mathematically derived from the geometric features of the Slovenian and Bosnian republic flags. The aim of this approach is to convey the state of human consciousness at a particular time and place whilst confronting an external objective reality.

In an interactive networked music-making environment, all participants, including the audience, have the ability to impact the performance flow, generate sound, and make music material

selections. Each performance of the same score might be different depending on the participants and the decisions and actions taken by them. In this democratised environment, the evaluation of compositional aesthetics encompasses not only the quality of the listening experience but also the quality and outcomes of active participation. The technical performance of the system, user front-end design, clarity of interactive features, and understanding of user action outcomes all impact the quality of the participants’ experience. Any aesthetic evaluation of networked music-making must consider the type and quality of relationships between different participants, as well as how well the performance outcomes match participants’ intentions.

From a compositional perspective, each composition should possess a clearly recognisable identity, regardless of who the performers are, the size of the audience, the audience members’ musical experience, the decisions and actions made during the performance, the venue’s size, dimensions, available technical resources, and other variables. Whilst every performance may differ both in aural and visual aspects, the score should be crafted in a manner that preserves the composition’s identity in each rendition.

Ultimately, the question is whether the networked composition and performance add value to and enhance the depth of participants’ experience compared to both the traditional one-directional performance flow and the superficial clickbait participation that is prevalent in contemporary culture.

3.1.3 Score Commentary

The inspiration for this composition came from the fact that Interface is a “Pierrot plus” ensemble (standard quintet + percussion). This fact prompted me to revisit Schoenberg’s seminal work *Pierrot Lunaire* (1912). Schoenberg’s composition is a setting of 21 poems from Otto Erich Hartleben’s German translation of Albert Giraud’s cycle written in French. Therefore, the piece inherited a process of material mapping (language translation) from its inception. The cycle structure split into three groups of seven poems is shown in Figure 3.1.

No/	Part One	Part Two	Part Three
1	<i>Mondestrunken</i> (Moondrunk)	<i>Nacht (Passacaglia)</i> (Night)	<i>Heimweh</i> (Homesickness)
2	<i>Columbine</i>	<i>Gebet an Pierrot</i> (Prayer to Pierrot)	<i>Gemeinheit!</i> (Vulgarity)
3	<i>Der Dandy</i> (The Dandy)	<i>Raub</i> (Theft)	<i>Parodie</i> (Parody)
4	<i>Eine blasse Wäscherin</i> (An Ethereal Washerwoman)	<i>Rote Messe</i> (Red Mass)	<i>Der Mondfleck</i> (The Moonspot)
5	<i>Valse de Chopin</i> (Chopin Waltz)	<i>Galgenlied</i> (Gallows Song)	<i>Serenade</i>
6	<i>Madonna</i>	<i>Enthauptung</i> (Beheading)	<i>Heimfahrt (Barcarole)</i> (Homeward Bound)
7	<i>Der kranke Mond</i> (The Sick Moon)	<i>Die Kreuze</i> (The Crosses)	<i>O Alter Duft</i> (O Ancient Fragrance)

Figure 3.1: *Pierrot Lunaire* poems (3 x 7)

As can be observed in Figure 3.1, the poem in the centre of the cycle (poem 4 in section Two) is *Rote Messe* (English translation: Red Mass). In the original context of the poem, the word “Mass” refers to a religious service (“At gruesome grim communion... Comes to the altar - Pierrot!”). The English word “mass” is a homonym and can be used to describe a large number of people crowded together. The first response that came to my mind upon hearing the phrase “Red Mass” was a childhood memory of large communist rallies with workers’ red flags and youths’ red scarves.

I followed the principle of taking the formal structural elements from the original poem as the basis of the piece, whilst transforming the material through the compositional process to reflect my personal reaction to the original material. Each *Pierrot Lunaire* poem consists of thirteen lines (two four-line verses followed by a five-line verse). The first line of each poem occurs three times (being repeated as lines eight and thirteen). This poem structure (rondel), visualised in Figure 3.2, is used as the basis for my piece *Red Mass*.



Figure 3.2: Pierrot Lunaire rondel poem structure

From the poem analysis and the fact that Schoenberg used numerology before the invention of twelve-note serialism, the recurring numbers 3, 7, and 13 were noted as structurally significant. As the first step in the pre-compositional structure definition, the duration of the piece was set to 7 minutes (420 seconds). The overall duration was then divided into 13 sections, each lasting approximately 32.3 seconds. As the poem’s rondel structure is effectively a ternary form, three distinctive composition parts were defined “Motion”, “Stillness”, and “Return”, as displayed in Figure 3.3.

	“Red Mass” ~ 7 min (420 sec)												
Part	Motion (~ 2:10 min)				Stillness (~ 2:10 min)				Return (~ 2:40 min)				
Section Id	1A	1B	1b	1a	2a	2b	2B	2A	3a	3b	3b2	3a2	3A
Tempo bpm	80	90	110	140	70	70	70	70	140	110	110	140	80

Figure 3.3: The structure of *Red Mass*

The tempo curve was defined so that in part one (“Motion”) the tempo increases in each consecutive section (80, 90, 110, 140 bpm), in part two (“Stillness”) the tempo is constant at the lowest value (70 bpm), and in part three (“Return”) each section is at the same tempo as the corresponding section from part one. Following the rondel cycle definition, I decided that only two basic compositional material types (“A” and “B”) would be used. Materials denoted as lower case ‘a’ and ‘b’ were to be derived from the corresponding main material. In subsequent sections where the type letters were repeated, the material was transformed and given a version number (eg A1, A2 and A3). Each section was then broken down into individual bars as shown

in Figure 3.4. The basic time signature for material A was set to 7/8 and for material B to 13/8 (subdivided into 8/8 + 5/8).

Section (32.3 sec)	bpm	beats	1/8s	bars	start bar No	end bar No
1A	80	43.0	86	11 * 7/8 + 9/8	1	12
1B	90	48.5	97	7 * 8/8 + 7 * 5/8 + 6/8	13	27
1b	110	59.0	118	8 * 8/8 + 8 * 5/8 + 2 * 7/8	28	45
1a	140	75.5	151	21 * 7/8+4/8	46	67
2a	70	37.5	75	10 * 7/8 + 5/8	68	78
2b	70	37.5	75	5 * 8/8+5 * 5/8+7/8 + 4/8	79	90
2B	70	37.5	75	5 * 8/8+5 * 5/8+7/8 + 4/8	91	102
2A	70	37.5	75	10 * 7/8 + 5/8	103	113
3a	140	75.5	151	21 * 7/8+4/8	114	135
3b	110	59.0	118	8 * 8/8 + 8 * 5/8 + 2 * 7/8	136	153
3b2	110	59.0	118	8 * 8/8 + 8 * 5/8 + 2 * 7/8	154	171
3a2	140	75.5	151	21 * 7/8+4/8	172	193
3A	80	43.0	86	11 * 7/8 + 9/8	194	205

Figure 3.4: *Red Mass* section info

The material “A” pitch set (Figure 3.5) was derived from the notes played in the opening beat of Schoenberg’s *Rote Messe* by all instruments. The normal form pitch set was constructed according to the pitch set theory (Forte 1973).



Figure 3.5: *Rote Messe*, first beat pitches as the chord and normalised pitch set

In contrast to the descending opening piano figure in *Rote Messe*, the ascending sequence played by Piccolo and Clarinet was constructed from the defined pitch set and used as the opening in *Red Mass* (Figure 3.6).

In the opening sequence, all vertical pitches sounding at the same time are either 1 or 2 semitones away from each other in the normalised pitch set form. The idea to use close intervals

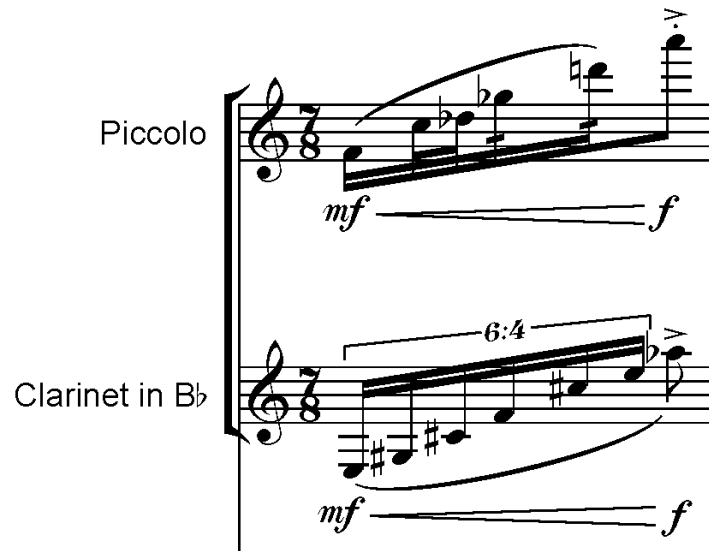


Figure 3.6: *Red Mass*, opening ascending sequence

(minor and major second and their inverted forms) was inspired by the chromatic movement of parallel fifths in the Cello part of *Rote Messe* as shown in Figure 3.7.



Figure 3.7: Cello part opening parallel fifths in *Rote Messe*

The material “A” piano chord (Figure 3.8) was also derived from the Cello opening dyads (Figure 3.7). Instead of using descending parallel fifths, I decided to subvert Schoenberg’s classical counterpoint subversion and move the base pitch up by a semitone, as shown in Figure 3.8. The resulting chord was split into two augmented fourths intervals and two semitone intervals, which were used extensively throughout the score (Figure 3.8).



Figure 3.8: *Red Mass*, material “A” piano chord and derived intervals

The rest of material “A” was derived either from the defined pitch set (Figure 3.5) or the piano chord (Figure 3.8). Most of the vertical pitch relationships use either minor second or augmented fourth intervals in the normal form pitch set.

For material “B”, the intention was to create a contrasting musical character to material “A” by using unstable pitched and unpitched sounds such as multiphonics, air notes, staccato, pizzicato, and tremolo. A different pitch set was defined for “B” material (Figure 3.9), also extracted from *Rote Messe*.



Figure 3.9: *Red Mass*, Material “B” pitch set

Other important influences in the *Red Mass* compositional process include the Sprechgesang (Wood 1946) singing technique from *Pierrot Lunaire* and Ganga (Petrović 2018), a traditional singing technique from rural Bosnia and Herzegovina. This recording of the female group (Crnačka Ganga 2014) illustrates the Ganga singing style. Both techniques use inflections of either continually rising or falling pitch. In Ganga singing, vocal groups of up to five singers follow the lead vocalist by pitching their entry a semitone above or below the lead vocalist. Singers then gradually move between minor second and unison intervals with microtonal inflections and individual timings. The glissando technique is used extensively in *Red Mass* to reflect the Sprechgesang and Ganga microtonal inflections.

3.2 Ukodus

For:

C flute

B \flat Clarinet

Violoncello

Piano

Duration:	ca. 3 minutes
Links:	Full Score ZScore Package Score follow video System overview and workshop video (7:25)

3.2.1 Objective: Linear Networked Composition And Performance

This piece was written in March 2017 for a workshop with the Moscow Contemporary Music Ensemble (MCME). *Ukodus* was the first composition written for the ZScore system, requiring the design and implementation of the entire networked composition and performance process, including score authoring, real-time notation scheduling and distribution, and dynamic notation view rendering on computer screens. For real-time event scheduling, distribution, and performance control over a network, I developed a proprietary ZScore server (2) and Control GUI (3.2.1.6) in Java. This chapter focuses on the notation authoring and dynamic rendering approach presented in Chapters 3.2.1.1 to 3.2.1.5. The core approach to dynamic notation implemented for *Ukodus* has been retained throughout this research and further refined in subsequent compositions. The MCME workshop served as valuable proof of concept for the ZScore system in a live performance environment.

3.2.1.1 Dynamic Notation

A composition data format produced by the score authoring tool needs to contain enough information to enable notation rendering tools to reliably reproduce the intended score layout and perform any time related operations. Semantic data models, such as GUIDO or MusicXML, define both spatial and temporal contexts for notation rendering. The problem with semantic representations is that both the score authoring tool and all participating notation rendering clients need to fully support the composer's intended notational style. Due to the vast variety of contemporary composition styles, extended playing techniques, and many contemporary composers' intentional disregard of standardisation, it would be hard to create a generic yet comprehensive semantic representation solution. As a result, composers and performers are increasingly turning towards systems which allow for constraint-free, graphic notation representations.

Modern computing utilises two major computer graphics forms: raster (bitmap) and vector graphics. The raster format defines actual pixel values to be displayed on the computer screen. It is fast to render, but the file size can grow significantly for higher resolution images, impacting network transfer timings. Additionally, raster format does not scale optimally, often resulting in visible quality loss when downscaled. As it is not human-readable, any change requires specialised editing tools.

On the other hand, vector graphics formats define shapes, paths, and their attributes such as colour, thickness, fill, etc. Although vector graphics formats may require additional interpretation time for rendering compared to raster formats, they offer superior quality scaling. Scalable Vector Graphics (SVG) (2001), one of the more popular vector graphics formats, enjoys widespread support from all major Internet browsers and numerous other graphical applications. SVG format is human-readable and can be extended with additional elements, attributes, scripts, and reusable custom symbols. This additional content may be used to create a musical context, opening up possibilities for music notation rendering, animation, and automation. Composers can create custom SVG symbol libraries that can then be distributed to networked clients and referenced in scores. This approach significantly reduces the amount of data that needs to be transferred in real-time.

As demonstrated by Gottfried (2015), the SVG format can be successfully transcoded to OSC. This capability has been integrated into the Odot library and, more recently, the Drawsocket notation system (Gottfried and Hajdu 2019). However, authoring more complex SVG structures, including mixed notation music scores, still requires specialised tools such as Adobe Illustrator (1987). Typically, scores are exported as SVG files offline ahead of a performance. With Adobe's recent announcement of support for Node.js, there might be potential for the network integration of Illustrator and real-time notation export in the future.

After extensive research of available networked dynamic notation tools and strategies, IN-Score standalone implementation was initially selected (in 2016) due to its networking capabilities, native OSC support, time-space mapping, built-in interactivity, and scripting engine. It also supports multiple graphics file formats, although its SVG support depends on the underlying Qt library (1995), so some features such as symbol referencing via the `xref` attribute were not available when *Ukodus* was created. To work around this, all score pages were exported and distributed in PNG raster format. The page graphics file sizes were optimised and exported for laptop computer screens used in a performance, avoiding scaling issues on the client-side. All required files were preloaded onto INScore clients to reduce network load and latency during a performance.

3.2.1.2 Alternating Pane Layout

The dynamic notation implementation in ZScore uses an innovative alternating pane dynamic notation update strategy. This strategy aims to resolve dynamic notation update issues by providing familiar left-to-right and top-to-bottom reading direction and ample preparation time to musicians. Furthermore, it defines allowed time windows for the upcoming notation generation and transfer, as well as the notation refresh timings that do not impact performance flow. The stave notation itself does not move whilst being read, however, several objects indicating tempo, current position, and conducting gestures are animated on top of the notation.

Figure 3.10 shows the main sections of the alternating pane layout for a full score and an instrumental part. In both cases, the notation view is divided into three main areas (panes). The

top pane contains information about the score (title, part name, server status etc.), actionable buttons (for interaction with the server or other peers) and signalling information (tempo and start indicators). The main area, which takes approximately 80% of the available screen space, is split into two notation panes, A and B. Each of the notation panes displays an equivalent of a full score page or a single part stave. The notation is read from left to right and top to bottom, the same as static paper scores.

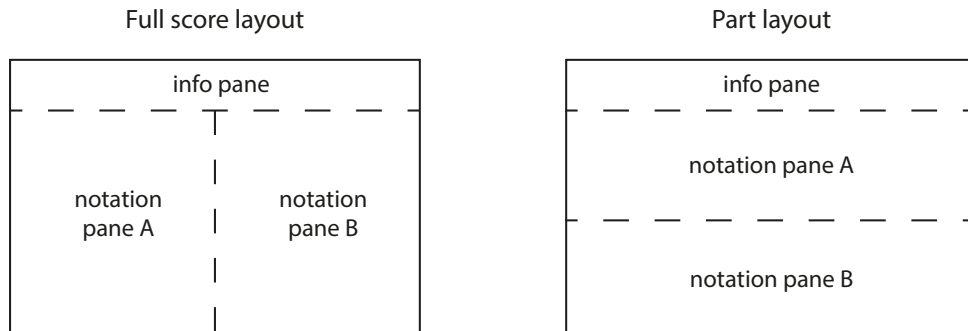


Figure 3.10: Alternating pane layout for a multiple stave full score and a single stave part

At any point of time during a performance, there is always one active and one preparatory pane. At the very start, pane A is active and pane B is preparatory. When the notation content in pane A is completed, pane B becomes active and pane A preparatory. Once the musician's focus is firmly moved to pane B, pane A is updated with the upcoming notation, which is scheduled to be performed after pane B's notation content is completed. The dynamic update process then continues in a similar fashion, following an ABAB... sequence.

There are several time restrictions that should be taken into account when working with alternating notation panes. In order to allow for performance continuity and sufficient preparation time for musicians, the notation to be played after the active pane notation is completed needs to be generated, transferred, and rendered in the preparation pane by the time the active pane notation is around halfway through its duration ($T/2$). In Figure 3.11, this time is marked as T_2 . Furthermore, the preparation pane notation should only be refreshed once the active pane notation is played for an appropriate time duration (T_1 , e.g. longer than one beat) to allow for the musicians' focus to switch. This means that the effective time window for notation preparation (generation, network transport, rendering, etc.) is from the active focus switch time (T_1) to the active pane half duration time (T_2). If, for example, the composition tempo is 120 bpm and the active notation pane contains 5 bars with a 4/4 time signature, then the notation preparation time window is 4.5 seconds ($T_1 = 0.5s$, $T_2 = 5s$). In most cases this would be sufficient time for the network transfer of graphical stave files or the generation of algorithmic notation. This also creates clear timeline rules for the real-time notation generation and display when using the alternating pane layout.

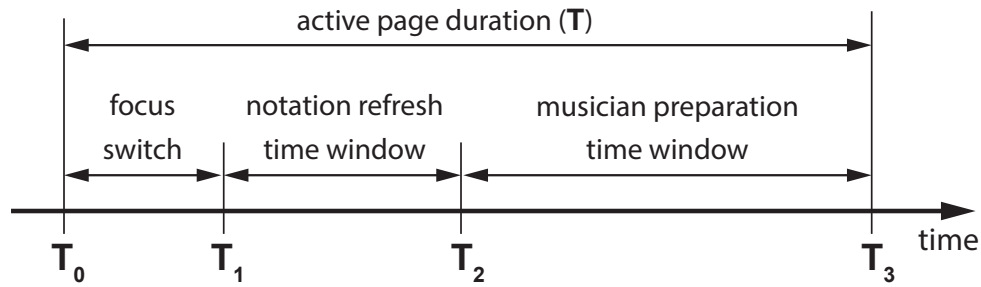


Figure 3.11: Alternating pane time windows

3.2.1.3 Time-space Mapping And Synchronisation

ZScore utilises the tempo-relative time synchronisation described above, which takes advantage of INScore’s time-space mapping functionality. In this mode, the master server application sends regular synchronisation messages carrying the global tempo-relative position to all clients. Each INScore client runs its internal clock and can synchronise independently if given a tempo and time-space mapping configuration. The master synchronisation events effectively override internal client clocks with the global tempo-relative position and, therefore, ensure common time-space positions across the clients. The synchronisation message frequency can be selected per composition and its choice depends on tempos and rhythmical structures used in the score.

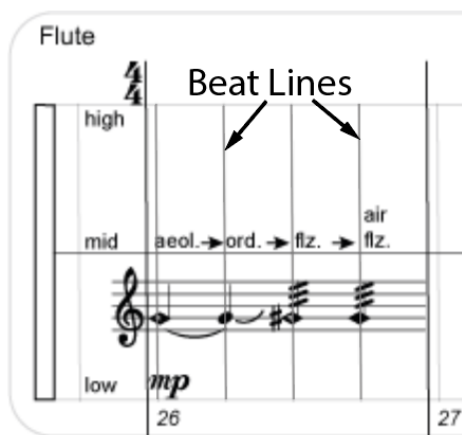


Figure 3.12: *Ukodus*, flute part excerpt illustrating Beat Lines

The concepts of a Beat Line (BL, Figure 3.12) and Beat Division Unit (BDU) were introduced for easier time-space mapping and event scheduling workflows. The BDU value can be set to any fraction of a whole note (e.g. 1/8, equivalent to a quaver duration) and represents the lowest time resolution available for event scheduling and synchronisation. The current minimum BDU value is 1/96. Beat Lines coincide with the bar beat onsets and contain information about their spatial and time position. The BL spatial position is set in terms of x and y coordinates on the score page, whilst their time position is expressed in a number of BDU units from the composition start. Beat Lines are a form of proportional notation; however, there are no restrictions regarding consecutive spatial Beat Line positioning, so they can be set individually

to suit the score notation density. The time interval between Beat Lines is measured in BDU units. For example, if the BDU value is set to 1/8, then in a 4/4 bar, each time interval between Beat Lines is 2 BDU units, and in a 5/8 bar with the beat division of 3+2/8, the first beat consists of 3 and the second beat of 2 BDU units. Beat Line spatial and time positions are exported with the score data and are used in INScore client for space-time synchronisation.

3.2.1.4 Notation Authoring

Functional requirements set out in Section 1.8 ask for a tool that can provide complex symbolic, graphic, or mixed notation authoring for any acoustic or electronic instrument, as well as the ability to export data in a suitable digital format for distribution over a computer network. After a significant research effort, I settled on a vector graphics editor, Adobe Illustrator (1987). It allows for the unconstrained positioning of any notation type, the export of SVG and multiple raster formats, the import and creation of user-defined symbol libraries, and it is scriptable, which opens a range of opportunities for functional extensions and potential real-time network integration. Illustrator does not provide any musical context by default, and therefore, a number of improvements have been implemented for more efficient music composition flows and integration with the ZScore software.

Inspired by Gottfried (2015), a hierarchical layer structure was created to provide musical context in Illustrator and enable the automation of score creation and export. The hierarchical layer elements can contain one or more child layers (Figure 3.13).



Figure 3.13: ZScore hierarchical score layer entity relationships

A Part layer has a one-to-one relationship to a Stave layer, which contains all the graphics data required to display an instrument staff. The Bar layer contains all the graphics data required to render the notation and logical data required for synchronisation, such as tempo, time signature, and beat line positions (Figure 3.14).

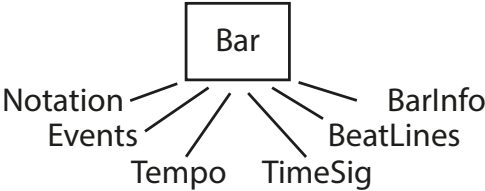


Figure 3.14: Bar layer content

The Notation layer contains all the symbolic or graphic data required to display bar notation and can contain arbitrary notation types. The Illustrator layer structure is displayed in the screen capture in Figure 3.15.

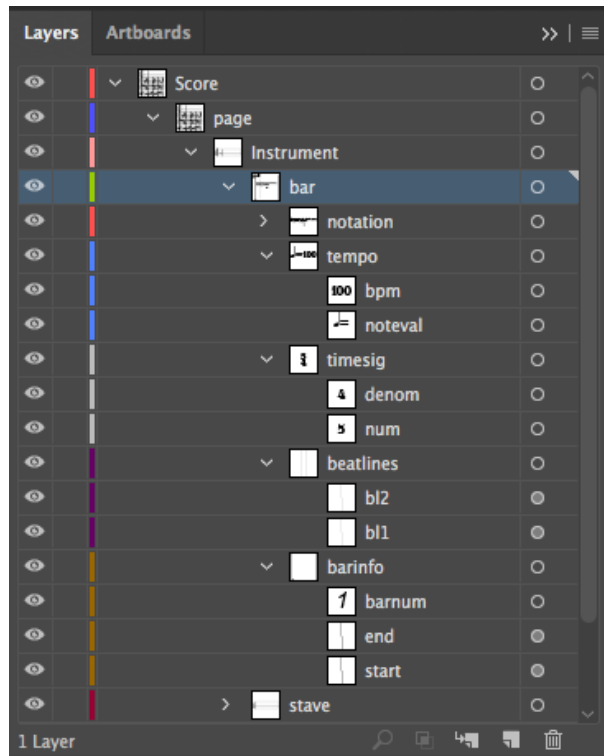


Figure 3.15: Adobe Illustrator layer structure

To accelerate symbolic notation generation, a set of custom symbols based on open-source LilyPond (1997) notation font were imported into Illustrator (Figure 3.16). Due to the flexibility of a vector graphic editor, it was straightforward to extend the library with custom symbols, such as different note head types and sizes and instrument fingering charts. For variable-length continuous lines, such as crescendo and decrescendo markings, a set of brushes were created and imported into Illustrator.

An example of mixed symbolic and graphic notation created in Adobe Illustrator with the help of ZScore tools plugin and music symbol libraries is displayed in Figure 3.18.

The set of JavaScript plugins were developed to speed up composition workflows and automate score export. Currently, the ZScore Tools plugin includes: Layer, Page, Bar, and Export management functionality (Figure 3.17). The Layer manager allows for Illustrator layer structure definition, editing, XML import/export, and copying between one or more scores. Similarly, Page and Bar managers help create required pages and bars at specified locations, including any metre or tempo changes and Beat Line positioning.

The Export manager allows for the export of the full score and parts in SVG or PNG graphics formats. In order to provide accurate and automated space-time mapping, the export process also creates necessary data for INScore in the required format:

$$([X_{start}, X_{end}]([Y_{start}, Y_{end}]([BB_{start}, BB_{end}])))$$

In this format, X and Y are space coordinates, and BB is the number of Base Beats expressed in

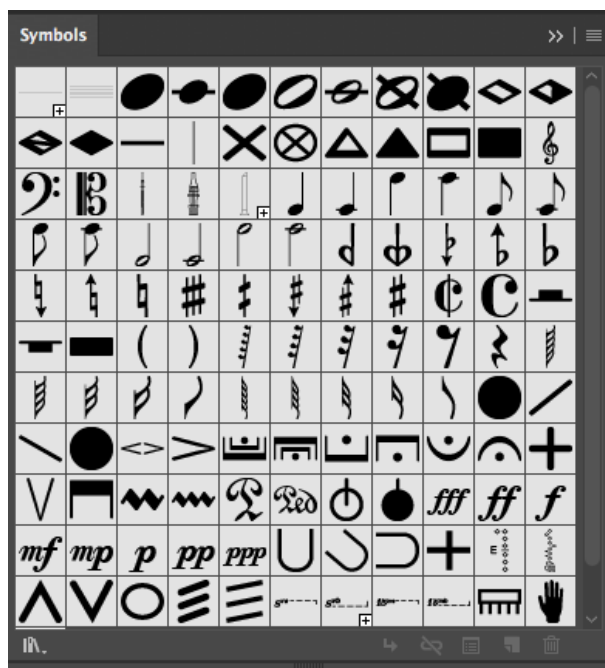


Figure 3.16: Notation symbol library for Adobe Illustrator

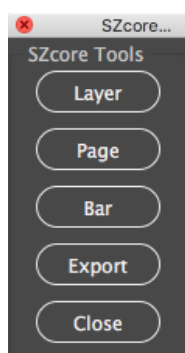


Figure 3.17: ZScore Tools plugin for Adobe Illustrator

BDU since the beginning of the section. The exported values define two-dimensional rectangles between two Beat Lines and the corresponding start/end tempo-relative time. Additionally, the export process automatically collates score metadata required by the score scheduling engine. This information about time signatures, tempo changes, other score events, BL positions, and related BB values is currently stored in a CSV (comma-separated values) formatted file.

Inspired by Helmut Lachenmann, all staves in *Ukodus* use experimental instrument-specific graphic clefs that allow for mixed sound and action notation. Figure 3.19 shows the flute staff for the example page displayed in Figure 3.18. The experimental clef indicates an approximate position within the instrument range where graphically notated sounds should be pitched. Three helper lines – low, mid, and high – indicate the bottom, top, and middle of the instrument range, respectively. The symbolic notation indicating specific pitches to play takes precedence over the graphic clef position. For example, in Figure 3.19, bar 33, the score asks the flautist to play a tongue ram using unspecified fingering at the very bottom of the flute range. However, in bar

Figure 3.18 shows a page of musical notation for the piece *Ukodus*, featuring mixed notation for four instruments: Flute, Clarinet B \flat , Cello, and Piano. The page is in 4/4 time and spans measures 31 to 35. The Flute part includes a vertical staff with high, mid, and low registers, and a standard musical staff with notes and dynamics like *p*, *mf*, *mp*, and *mf*. Annotations include 'ord.', 'aeol.', 'air flz.', 'tongue ram', and 'air aeol. ord.'. The Clarinet B \flat part also has a vertical staff and a musical staff, with dynamics *mf*, *mp*, and *mf*, and annotations like 'ord.', 'flz.', and 'slap tongue'. The Cello part features a vertical staff with a cello icon and a musical staff, with dynamics *mp*, *mf*, and *mp*, and annotations like 'circular bowing', 'harmonic gliss', 'behind bridge', 'col legno batutto jeté', 'arco', 'msp', and 'harm gliss'. The Piano part includes a vertical staff with a piano icon and a musical staff, with dynamics *p*, *mf*, and *mp*, and annotations like 'both hands on strings rasgueado', 'RH mute on harmonic', and 'finger hit on strings'. The page is labeled 'P8' in the top right corner.

Figure 3.18: *Ukodus*, an example of the score page containing mixed notation

35, the flautist has to play a bottom octave G, transitioning from an airy to a fully pitched timbre.

The clarinet stave (Figure 3.20) is similar to the flute stave, indicating the approximate instrument range position. The outline of the clarinet shape in the clef does not hold any musical meaning, so it was not reused in any of the subsequent scores. The principles of symbolic notation precedence and unspecified pitch notation are identical to the flute part.

Figure 3.19 is a close-up of the flute stave from Figure 3.18. It shows the Flute part in 4/4 time, measures 31 to 35. The staff is divided into high, mid, and low registers. The musical notation includes notes, rests, and dynamics: *p*, *mf*, *mp*, *mf*, and *p*. Annotations include 'ord.', 'aeol.', 'air flz.', 'tongue ram', and 'air aeol. ord.'. The page is labeled 'P8' in the top right corner.

Figure 3.19: *Ukodus*, flute stave example

The cello clef in *Ukodus* (Figure 3.21) is very similar to Lachenmann's clef in *Pression* (Figure 1.4). The horizontal lines extending from the clef indicate the approximate playing position, with two dashed lines showing the end of the fingerboard and the top of the tailpiece,

Figure 3.20: *Ukodus*, clarinet stave example

whilst the solid line indicates the bridge position. The notation at the beginning of the excerpt in Figure 3.21 calls for the precise left hand position, whilst the approximate bow position is indicated by the line above the symbolic notation. The bow, in this case, moves gradually between *sul tasto* and *sul ponticello* positions, ending up in a circular bowing movement.

Figure 3.21: *Ukodus*, cello stave example

In bar 33, the cellist is instructed to perform Bartok pizzicato on string IV whilst executing a downwards glissando with the left hand using harmonic finger pressure. In bar 34, the score specifies that the right hand should first play pizzicato behind the bridge on string IV and then transition to executing *col legno battuto jeté* on the same string, still positioned behind the bridge. In the final bar of the page, the cellist plays a harmonic glissando with the left hand whilst the bow moves gradually from *molto sul ponticello* to *sul tasto*.

The experimental piano staff in *Ukodus* (Figure 3.22) is vertically split into two areas by a line coming from the top of the keyboard. The lower area is designated for the notation of pitched material and any actions performed on the keyboard, whilst the upper area is reserved for actions taking place directly on the strings, inside the piano.

On page 8, as shown in Figure 3.22, the pianist is initially asked to play bottom A and then, with the sustain pedal engaged, continually strum the strings with fingers (*rasgueado*) within the pitch area indicated by the black curvy shape on the stave. The height of the black area indicates an approximate range of strings that should be affected by strumming.

In the last beat of bar 33, the pianist is instructed to hit strings with the palm of their right hand and then mute the bottom A string, ideally at the position where a harmonic can be heard. At the same time, the left hand plays the A key on the keyboard, followed by silently playing

Figure 3.22: *Ukodus*, piano stave example

a minor third dyad by gently pressing the keys, avoiding the hammer hits. Whilst holding the dyad keys depressed with the left hand, the fingers on the right hand hit the strings directly in a triplet rhythm, causing the open strings to vibrate.

3.2.1.5 Notation Rendering

Figure 3.23: *Ukodus*, cello part dynamic notation in INScore with alternating pane layout

Two INScore client versions (part.inscore and full.inscore) were created, one for the full score vertical alternating pane and one for the horizontal part alternating pane layout. These startup files contain layout configuration and the set of JavaScript functions that handle all interactive tasks. Once the INScore startup file is opened, all communication with the server

can be done directly from the INScore client.

An example of the dynamic score with alternating pane layout is presented in Figure 3.23. The active notation pane is highlighted with red borders, whilst the preparatory pane is slightly dimmed. Several features that aim to replace some of the conducting gestures have been added, such as the signalling traffic lights at the top left corner, which draws the attention of musicians and provides an indication of the starting tempo. The traffic lights are used only at the beginning of the play as a replacement for the conductor's upbeat gesture.

Additionally, the current position line is visible as the light green line on the upbeat leading to bar 28 (Figure 3.23). It has an attached tempo indicator, which is visible as the red circle on top of the staff. The actual play start position is marked with a light purple line on the first beat of bar 28 in Figure 3.23. The starting position can be set to any Beat Line from the network management client (Figure 3.25).

When the score performance is started in the Performance Control GUI (Figure 3.25), the traffic light signal flashes at the starting tempo frequency, and the position line begins moving from left to right, indicating the current position in time on the screen. The attached tempo indicator ball starts simulating conductor signals with vertical movements calculated from the simplified pendulum motion formula, where the ictus plane is at the top of the staff.

Figure 3.24: *Ukodus*, full score in alternating pane layout

The current position line always starts from the upbeat before the selected starting position, mimicking the familiar conductor gesture at the start of play. When the position line reaches

the penultimate beat of the active pane stave, the preparatory pane's current position line starts moving from its upbeat position at the same time, providing notation view continuity.

The full score view, shown in Figure 3.24, implements a vertical alternating pane layout (Figure 3.10). It is synchronised in the same way as other parts, including the traffic lights, starting and current position lines, and the tempo ball indicator.

3.2.1.6 Performance Control

The central hub in charge of scheduling and distributing the score data over a network in real-time is the ZScore server and management client written in Java programming language. The ZScore Performance Control GUI client (Figure 3.25) can import and parse score definition data exported from the ZScore Tools Illustrator plugin and submit to the server, creating an internal representation of the score metadata. The internal score object-oriented model mimics the layer hierarchy shown in Figure 3.13.

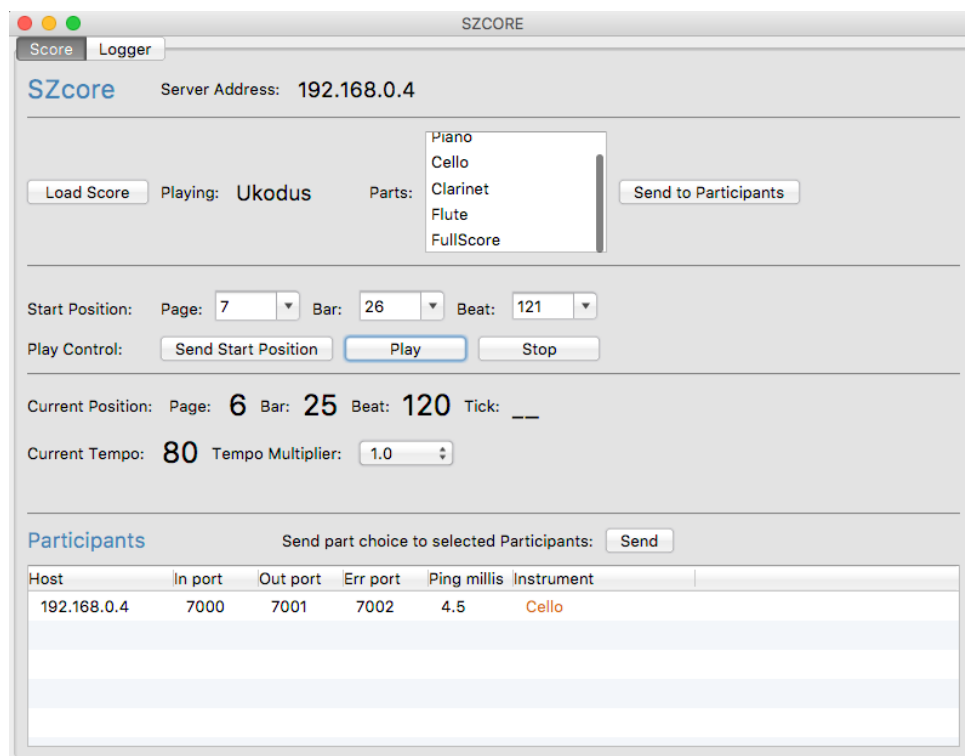


Figure 3.25: *Ukodus*, ZScore Performance Control GUI

The server listens to notation client connections and sends information about available parts to all the connected clients. When musicians select individual parts on their notation client, the server associates the selected score part with the client's host address. From that point on, all messages related to a particular part will be routed to the associated client address.

The server internal scheduling resolution is 1 millisecond, with a maximum measured deviation of 0.8 milliseconds. It translates local absolute time to a tempo-relative value expressed in BDU units and evaluates any scheduled score events accordingly. The server supports multiple

transports with different metres and tempos, allowing for compositions and performances of polyrhythmic and polymetric scores. Events can be pre-loaded in score data or dynamically added during a performance, adhering to the timing rules discussed above. ZScore's real-time functionality is bound by these timing rules, so any notation generated during the performance needs to be available in the time window defined by the T_1 and T_2 boundaries (Figure 3.11).

The Performance Control GUI (Figure 3.25) can load and start the score from any position defined in terms of the Page, Bar, and Beat number on all participating networked clients. It also allows for tempo multiplication in the range from 0.1 to 2.0, useful for rehearsal purposes, as well as for finely tuned dynamic tempo changes during a performance when required.

3.2.2 Score Commentary

The inspiration for *Ukodus* came from a particular Sudoku puzzle containing visually and numerically interesting symmetrical structures with multiple lines of symmetry. The name of the piece is the word 'Sudoku' reversed.

Sudoku is a logic-based, combinatorial number-placement puzzle. The objective is to fill a 9 x 9 grid with digits so that each column, each row, and each of the nine 3 x 3 sub-grids contain all of the digits from 1 to 9. The puzzle used for the composition structure of *Ukodus* is visualised in Figure 3.26.

From Figure 3.26, it is evident that the initial setup for this particular Sudoku puzzle displays the symmetrical positioning of numerical values in both horizontal and vertical directions. Further explorations of the relationships between the visual and numerical elements of the puzzle formed the basis for the compositional structure of *Ukodus*.

In order to quantify the positions of the puzzle's initial numerical values, each row and column in the puzzle was given a sequence id in the range [0, 4], as visible in the top row and first column of Figure 3.26. Sequence id 0 was assigned to the central row and column. The upper contour of the puzzle in the West \rightarrow East (W \rightarrow E) direction was then extracted as a graph (Figure 3.27). The vertical values in the graph indicate the distance from the central row.

Actual numerical values of the puzzle's upper area in the W \rightarrow E direction were also plotted as a graph in Figure 3.28. As the upper area columns can contain up to two numerical values, separate curves were defined for the inner and outer numerical values.

Due to the horizontal symmetry, the lower area of the puzzle in the W \rightarrow E direction has the same outer contour as the upper area. However, the actual numerical values are different, as displayed in Figure 3.29. Following the same approach, graphs for the vertical contour in the North \rightarrow South (N \rightarrow S) direction and their corresponding numerical value graphs were extracted and plotted.

The basic structure of the piece was derived from the obtained graphs and resulting data sets (Figure 3.30). The composition was split into nine sections, reflecting sudoku's intrinsic divisions. The tempo and the length of each section were calculated from the upper contour

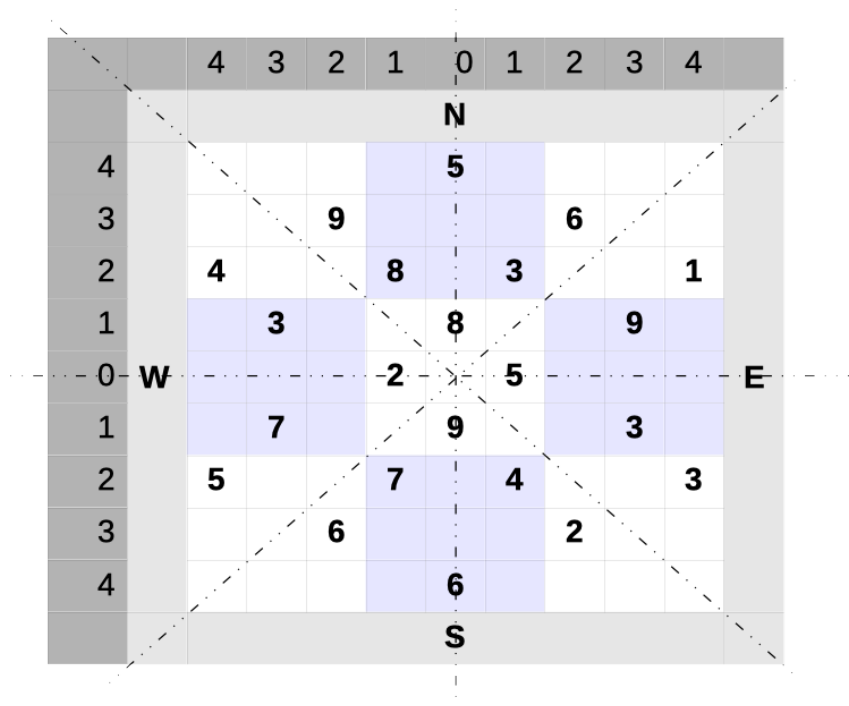


Figure 3.26: *Ukodus*, Sudoku puzzle used for the composition material

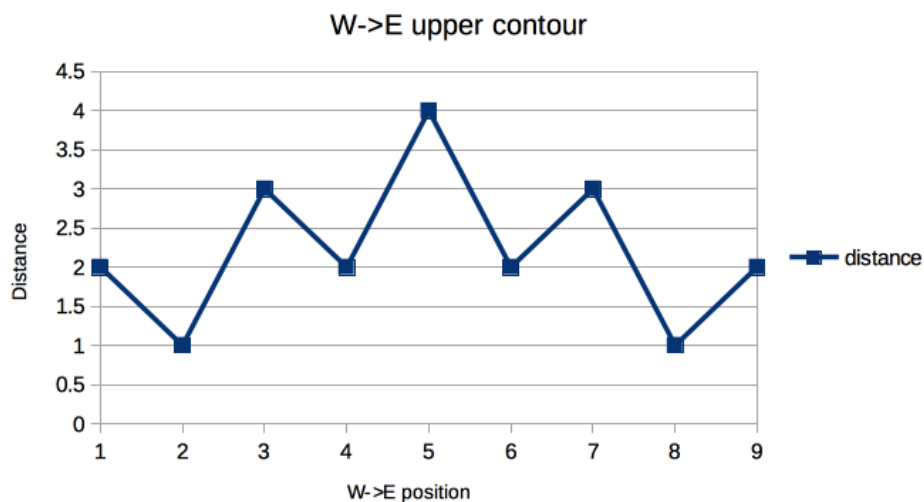


Figure 3.27: *Ukodus*, Sudoku puzzle W → E upper contour

values (Figure 3.27). The starting tempo value was set to 100 bpm, whilst the vertical unit value was set to 20 bpm. As can be deduced from the graph (Figure 3.27), the vertical contour value in the W → E direction decreases by one unit from position 1 to position 2. Therefore, the tempo for section 2 can be calculated using the following formula:

$$\text{starting tempo} - \text{direction} * [\text{number of steps} * \text{tempo unit value}] = \text{new tempo}$$

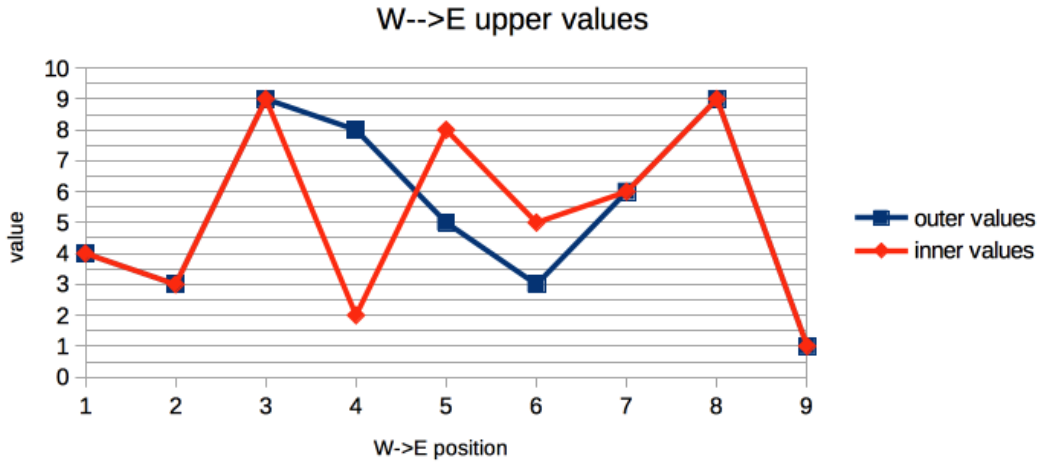


Figure 3.28: *Ukodus*, Sudoku puzzle W → E upper values

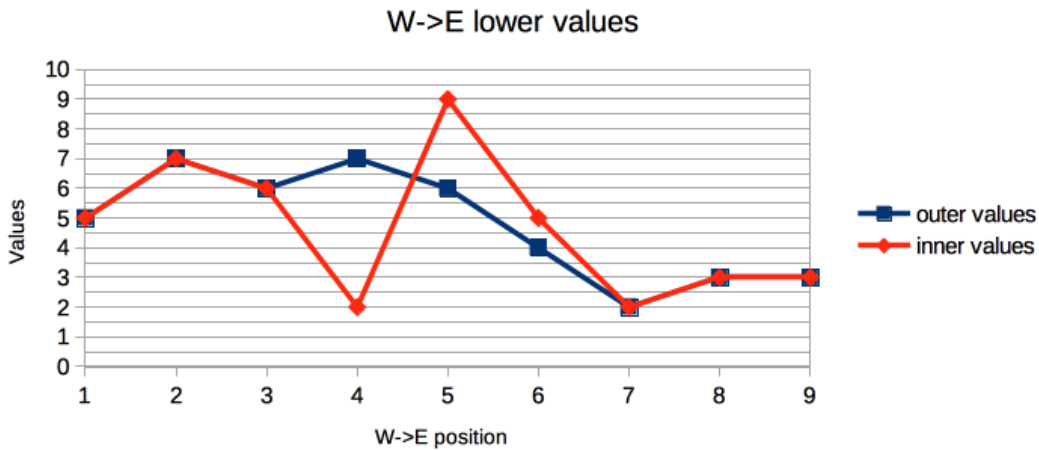


Figure 3.29: *Ukodus*, Sudoku puzzle W → E lower values

as:

$$100 - 1 * [1 * 20] = 80$$

The tempo and number of beat units for each section were then calculated using the formula above. The pitch range for each section was selected to reflect the same contour curve, starting with the mid-range for each instrument.

section	1	2	3	4	5	6	7	8	9
tempo bpm	100	80	120	100	140	100	120	80	100
beats (8ths)	100	80	120	100	140	100	120	80	100
range	mid	low	high	mid	very high	mid	high	low	mid
material	air/noise	pitch/noise	pitch	air/pitch	harm/mphon	air/pitch	pitch	pitch/noise	air/noise

Figure 3.30: *Ukodus*, basic structure

By following similar rules, each section was further divided into nine parts. The length of each part was calculated from the section's length and the upper contour (W → E) value for

each step. Therefore, the part's length is relative to the contour value for the equivalent position; the higher the contour value, the longer the part, and vice versa. The structure of section 1 is displayed in Figure 3.31. Here, each part is defined by its length in terms of the number of beat units (1/8), the division of beat units into bars, and the number of musical events based on the $W \rightarrow E$ upper values graph (Figure 3.28).

Distance sum	20									
part	1	2	3	4	5	6	7	8	9	9
distance/bars	2	1	3	2	4	2	3	1	2	2
Section 1	register	mid	material	air/noise	W->E upper val					
beats	10	5	15	10	20	10	15	5	10	10
divisions	5+5	5	5+5+5	5+5	5+5+5+5	5+5	5+5+5	5	5+5	5+5
no events inner	4	3	9	2	8	5	6	9	1	1
no events outer	4	3	9	8	5	3	6	9	1	1
bars no	1, 2	3	4, 5, 6	7, 8	9, 10, 11, 12	13, 14	15, 16, 17	18	19, 20	

Figure 3.31: *Ukodus*, section 1 structure

The other sections were defined in a similar way. The structures of sections 2 and 3 are displayed in Figure 3.32. All sections use different curves for event and material generation. For instance, section 2 uses the $W \rightarrow E$ lower value graph, whilst section 3 uses the $W \rightarrow E$ upper values in retrograde. The number of events in this context represents a guideline for the density of the musical material in each part.

Section 2	register	low	material	pitch/noise	W->E lower val					
beats	8	4	12	8	16	8	12	4	4	8
divisions	4+4	4	4+4+4	4+4	4+4+4+4	4+4	4+4+4	4	4+4	4+4
no events inner/pitch class	5	7	6	7	6	4	2	3	3	3
no events outer/pitch class	5	7	6	2	9	5	2	3	3	3
bars no	21, 22	23	24, 25, 26	27, 28	29, 30, 31, 32	33, 34	35, 36, 37	38	39, 40	
Section 3	register	high	material	pitch	W->E upper retrograde					
beats	12	6	18	12	24	12	18	6	12	12
divisions	3+3+3+3	3+3	6 * 3	4 * 3	8 * 3	4 * 3	6 * 3	3 + 3	4 * 3	4 * 3
no events inner/pitch class	1	9	6	5	8	2	9	3	4	4
no events outer/pitch class	1	9	6	3	5	8	9	3	4	4
bars no	41 - 44	45, 46	47 - 52	53 - 56	57 - 64	65 - 68	69 - 74	75, 76	77 - 80	

Figure 3.32: *Ukodus*, sections 2 and 3 structure

The instrumentation – consisting of Flute, Clarinet, Cello, and Piano – was selected to reflect the symmetrical nature of the material and to explore possibilities for flexible grouping and mirroring. For example, in section 1, the instruments were split into two groups: Group 1 (Flute and Clarinet), and Group 2 (Cello and Piano). Group 1's event density was extracted from the outer curve, whilst Group 2's event density was obtained from the inner curve.

The beginning of the piece (section 1, part 1) for Group 1 (Figure 3.33) illustrates how the extracted data is mapped to musical material. The number of events for section 1, part 1, can be looked up from Figure 3.31 – for both Group 1 and Group 2 it is four. The material for section 1 (Figure 3.30) is set to “air/nose”. Therefore, four musical events were created for both instruments in Group 1 (Flute and Clarinet). Internally, instrument lines within the group were constructed as a mirrored response/call whilst the shape of both gestures resembles the $W \rightarrow E$ upper contour (Figure 3.27).

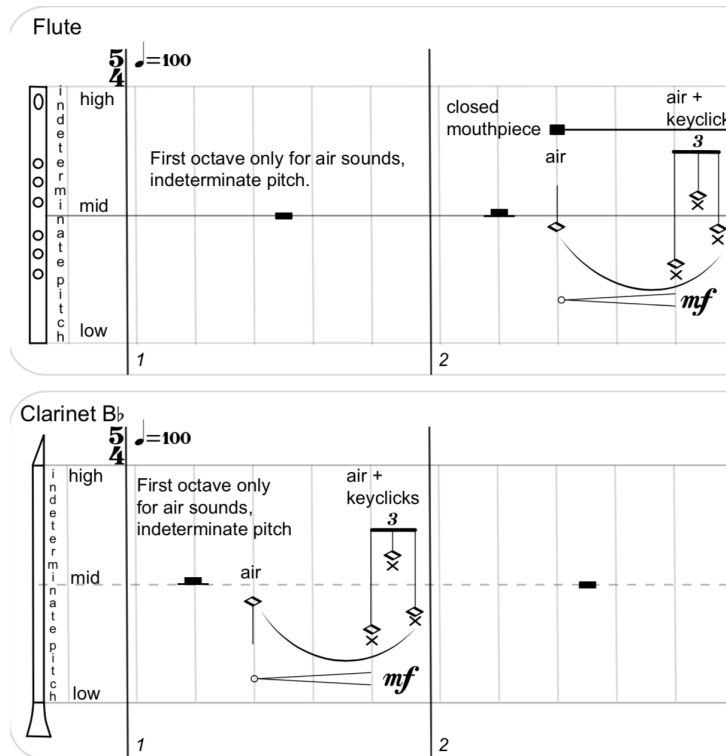


Figure 3.33: *Ukodus*, Group 1 events in section 1 part 1

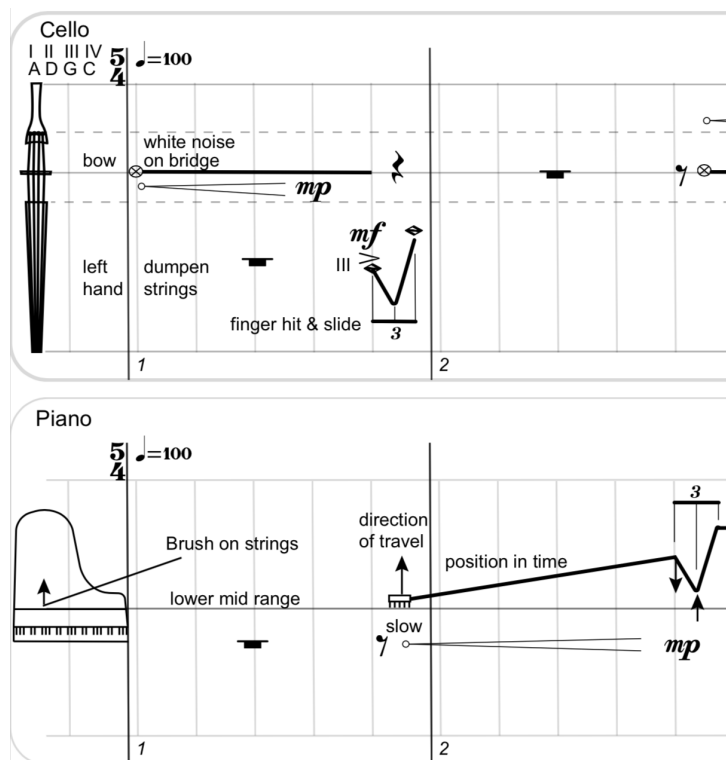


Figure 3.34: *Ukodus*, Group 2 events in section 1 part 1

Likewise, four “air/noise” events have been created for each instrument (Cello and Piano) in Group 2 for section 1, part 1. These events serve as a call/response mechanism. The Group 2 gesture shape reflection forms a symmetrical response to Group 1, as shown in Figure 3.34.

The rest of the material was created following a similar procedure. Due to the extensive work required for the technical aspects of ZScore software development and Adobe Illustrator score authoring, only the first 64 bars of the piece were completed in time for the MCME workshop.

3.3 Vexilla

For:

Bass Clarinet

Violin

Violoncello

Video Projector

Duration:	ca. 7 minutes
Links:	Full Score ZScore Package Score follow video with audience score visualisation Performance recording

3.3.1 Objective: Audience Score Visualisation

This piece was written for ZScore in early 2018 and was performed on 1 May 2018 at Deptford Town Hall by Heather Roche (Bass Clarinet), Valerie Welbanks (Cello), and Patrick Dawkins (Violin) thanks to Goldsmiths Graduate School and Music Department funding.

Vexilla were flag-like objects used as a military standard by units in the ancient Roman army (Figure 3.35). Later on, the meaning of the word extended to represent any object, such as a relic or icon, carried as a standard into battle. This piece explores the emotional and rational responses to various flags I have encountered throughout my lifetime, from fervent teenage patriotism to deep distrust of any ceremony that requires enthusiastic flag waving.



Figure 3.35: Roman Vexillum⁵

The intention behind the audience score visualisation was to represent the composition's ideas in an easily comprehensible format through various flag shape animations, inspired by the *Vexilla* music material described below. To achieve this, a canvas resembling a Roman vexillum was placed on the stage (Figure 3.36), clearly visible to all audience members. All score animations were projected onto the canvas from a dedicated laptop running INScore standalone

⁵Photo by Wolfgang Sauber, Roman Museum, Petronell (Lower Austria), Vexillum of the Ala I Thracum Victrix (replica), source: https://en.wikipedia.org/wiki/Vexillum#/media/File:Museum_Petronell_-_Vexillum_Ala_I.jpg (last visited 5 Sep 2023)

client. INScore (v1.21) was chosen at the time of score creation for its network connectivity features, scripting, and SVG graphics rendering ability.



Figure 3.36: *Vexilla* staging with synchronised animations projected onto a canvas

Embedded within the Adobe Illustrator score, the audience part includes scripts that control audience-specific visualisations synchronised with other musical events defined in the score. This innovative integration of the audience part within the score, later adapted for general audio-visual score elements, is further detailed in Chapters 3.3.1.1 and 3.3.1.2. Additionally, *Vexilla* introduces a colour-coding system for specific playing techniques, as further explained in Chapter 3.3.2.1.

3.3.1.1 Embedded Scripting

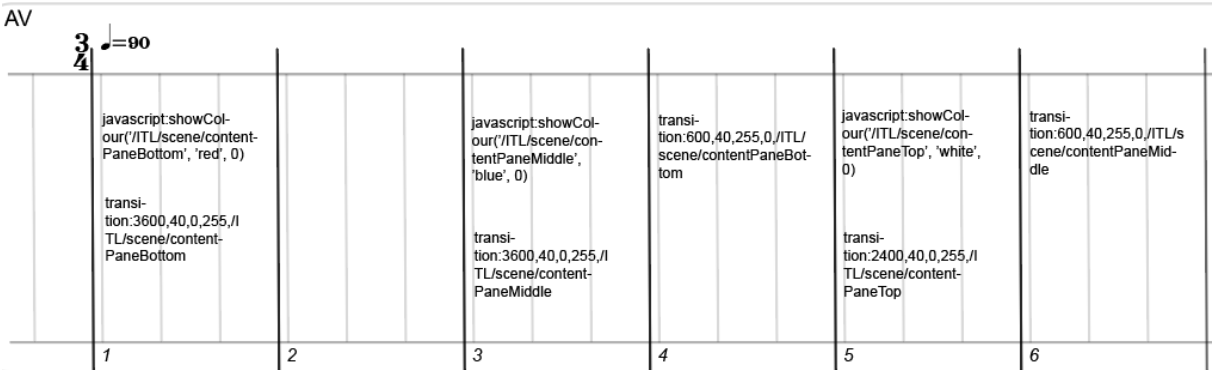


Figure 3.37: *Vexilla*, AV part

Early in the composition process of *Vexilla*, I opted to integrate commands for real-time audience visualisations directly within the score. Visual aspects were integral to the creative

process, demanding tight synchronisation with the musical material. To facilitate this, a dedicated AV (Audio/Visual) part was created in the Adobe Illustrator score (Figure 3.37) to serve as a container for any information related to audience score representation. The AV part is structurally identical to all other parts in the score, sharing the same bar and beat structure, tempo changes, etc.

The task of audience score visualisation was divided into three components: score event definition, server-side event scheduling, and INScore client SVG element configuration and rendering. In Adobe Illustrator’s layer structure, embedded scripts were placed inside the “events” layer, which was nested within the corresponding bar (Figure 3.38). In *Vexilla*, the event scheduling interval for embedded scripts is fixed to one bar, ensuring that the events’ execution always occurs on the first beat of the bar.

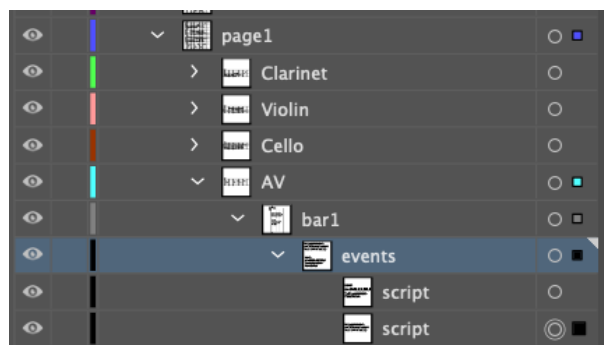


Figure 3.38: Adobe Illustrator layer structure for embedded scripts

Two types of events were created for the purpose of audience visualisation: “javascript” and “transition”.

INScore’s ability to execute arbitrary JavaScript embedded in OSC messages was extensively utilised in the implementation of the score visualisations. A generic “javascript” type event was defined to act as a simple carrier of the scripts defined in the AV part. All graphical elements and their addresses referenced in “javascript” events were defined in the `avPart.inscore` configuration file. The INScore instance responsible for the score visualisation needs to load the `avPart.inscore` file and register itself as the AV destination with the ZScore GUI to receive scripting events.

A script embedded in the score has a general form:

```
<eventType> : <script-content>
```

For example, in the Adobe Illustrator source file `Vexilla.ai`, an embedded script with “javascript” event type from the AV part, bar 1, contains:

```
javascript:showColour('/ITL/scene/contentPaneBottom', 'red', 0)
```

The script content – “`showColour('/ITL/scene/contentPaneBottom', 'red', 0)`” – is packaged in the OSC event and sent to INScore where it is executed on receipt. The function “`showColour`” referenced in the script is defined in the `avPart.inscore` file:


```
function showColour(component, colourName, alpha){...}
```

The “transition” type events manage SVG element alpha values. It quickly became apparent that there was a need for the generic opacity management of SVG elements, as a sudden appearance or removal of SVG elements resulted in visually unappealing effects that did not match the nature of the music material. A “transition” event can target any SVG element by referencing the element’s INScore address. The scheduling of the events is performed by the server based on the event configuration specified in the AV part.

An example of the “transition” event type from the AV part, bar 1, contains:

```
transition:3600,40,0,255,/ITL/scene/contentPaneBottom
```

This script gradually increases the opacity of the component named “contentPaneBottom” from completely transparent (alpha value 0) to fully opaque (alpha value 255) over a duration of 3600 milliseconds (3.6 seconds). All transition scripts control the alpha value of the named component.

On the server-side, this configuration is translated into an instance of the Java “Transition” interface using the following mapping:

```
transition: <duration>,<frequency>,<startValue>,<endValue>,<component>
```

Transition object instances are scheduled on the server based on the embedded script’s duration and frequency settings. The alpha value is calculated and sent during playtime inside the Java “TransitionScriptEvent” object. This event object is then translated into an OSC message that executes “setAlpha” JavaScript function in INScore:

```
function setAlpha(component, alphaValue){...}
```

3.3.1.2 Audience Score View

The content of audience visualisation elements is based on the same tabular structure (Figure 3.42) that is used to derive the music material and is, therefore, an integral part of the composition process. All graphical elements used in audience visualisation are defined in the avPart.inscore file. Each graphical element has a unique address. For example, the SVG element representing the UK flag has the address “/ITL/scene/ukFlag” and is defined in the avPart.inscore file as:

```
/ITL/scene/ukFlag set file 'rsrc/UkFlag.svg';  
/ITL/scene/ukFlag x 0.0;  
/ITL/scene/ukFlag y 0.0;  
/ITL/scene/ukFlag scale 1.66;  
/ITL/scene/ukFlag z 3.0;
```

In this instance, the content of the SVG element is associated with the file “UkFlag.svg” placed in the “rsrc” directory. The content of this element can be changed by calling the generic “setFile” JavaScript function defined in the avPart.inscore file:

```
function setFile(component, filePath) {  
    post(component, SET, FILE, filePath);  
    showComponent(component, 1);  
}
```

This function can be used to change the content of any INScore element that is using a file reference. Another approach would be to use INScore’s native OSC support and directly target the “/ITL/scene/ukFlag” address in an OSC message containing the required parameters. This approach would avoid JavaScript interpretation in the INScore client and, therefore, be slightly faster to render. However, INScore elements can have a number of parameters, as illustrated above, so changing them individually would require either a separate message for each parameter or an OSC message bundle containing all the required changes. In both cases, it would be more efficient in terms of network load to send a single function call that changes all the required parameters at once. A decision was made early on in the process to abstract the OSC level calls into higher-level functions to reduce both the number and size of required messages passed between the server and client.

All resources, including graphics files, are preloaded onto the laptop responsible for audience visualisation ahead of a performance, thereby avoiding any latency-sensitive network traffic during the score play. Furthermore, events that manage a graphical element’s appearance only contain references, such as the element’s address, function name, colour id, etc., rather than full graphics or complete definitions of values that need to be modified. All animated features of the audience score view are handled by a set of JavaScript functions embedded within the avPart.inscore file.

3.3.2 Score Commentary

Structurally, the composition is divided into three main sections: Yugoslavia (1945 - 1992), Europe (1992 - 2018), and Epilogue. Each main section is then further broken down into smaller parts. Section 1 (Yugoslavia) is split into seven parts, with six representing the former Yugoslav federative republics’ flags, and one for the main SFR Yugoslavia state flag. Section 2 (Europe) is divided into three parts (the EU, New Bosnia, and UK flags) whilst section 3 (Epilogue) contains only one part and references all flags used in the piece (Figure 3.39).

Each main section’s duration is less than half of the previous section, symbolising the perceived acceleration of the passage of time with age. However, each part within the main sections has the same duration (35.2 seconds), reflecting the equal importance (or unimportance) of flags in my life. Originally, the plan was to write a 10-minute piece split into 17 parts of

Section 1						
Yugoslavia, childhood, military, communist, nostalgia, friends, love, romantic						
Slovenia	Croatia	Srbija	Macedonia	Montenegro	Bosnia & Hzg	Yugoslavia
1	2	3	4	5	6	7
Section 2						
Europe, family, job, contemporary, capitalism						
EU	New Bosnia	UK				
8	9	10				
Section 3						
Epilogue						
Recap						
11						

Figure 3.39: *Vexilla*, Structure

equal duration, with each part lasting 35.2 seconds. However, due to time pressures, the overall number of parts was reduced to 11, whilst the individual part duration remained unchanged at 35.2 seconds.

The tempo of section 1 is set to 90 bpm, being the same as the Yugoslav national anthem used as the source for the compositional material in part 7. The tempos of sections 2 and 3 are set to 120 bpm, inspired by the official EU “Anthem to Europe”, which is based on Beethoven’s *9th Symphony* fourth movement (“Ode to Joy”). Part 10, dedicated to the UK, is inspired by Led Zeppelin’s “Immigrant Song”.

As a starting point, I documented my immediate factual and emotional responses to each section and part description (as displayed in Figure 3.39). It became evident right away that my responses to the first section were more centred around memories and nostalgia, whilst the second section revolved around the realities of living, survival, and the daily grind of life. To further inspire the composition, I also tapped into my associative memory, noting the first song or melody that came to mind when thinking about a particular flag. Extracts from these songs and melodies were then utilised to construct the compositional material for each part.

The common overarching thread across all composition parts is a personal reflection on the processes of othering, belonging, and separation. The process of othering is frequently used in political discourse where “others” are defined as different from “us”, representing a physical or economic threat to be feared. Almost invariably, othering rhetoric involves mythology and symbolism, including flags, serving to establish a distinction between “us” and “others”.

Another important process utilised in the generation of compositional material is the decomposition of flags into graphical facets and their mapping to sound categories and gestures. For example, the SFR Yugoslavia tricolour flag was composed of red, blue and white stripes with the socialist red star in the middle (Appendix C). These flag features are then mapped to sound categories as illustrated in Figure 3.40.

Additionally, the position of individual stripes on the flag is mapped to the instrument range in a vertical order; for instance, the bottom stripe is associated with the lower instrument register, and so on. Angled lines on flags are mapped to the pitch movement, so if a line is sloping

Flag visual feature	Sound category
Red colour	Warm, fully pitched sound
Blue colour	Cold, harmonic and multiphonic sounds
White colour	Air noise, percussive non-pitched sounds
Star	Pointillist glissandi gestures resembling the shape of a star

Figure 3.40: *Vexilla*, flag feature mapping to sound

downwards when looking left to right, the sound starts at a higher register and gradually slides down to a lower register.



Figure 3.41: *Vexilla*, *Na Planincah*, traditional folk song from Slovenia

Figure 3.42 illustrates the principles of the generation of compositional material in this piece. Part 1 represents the flag of Slovenia (Figure 3.39). Following the associative memory approach described above, the first song I could recall when thinking about Slovenia was the simple traditional folk melody “Na Planincah” (Figure 3.41), which happens to be one of the first melodies I learned to play as a child on the piano.

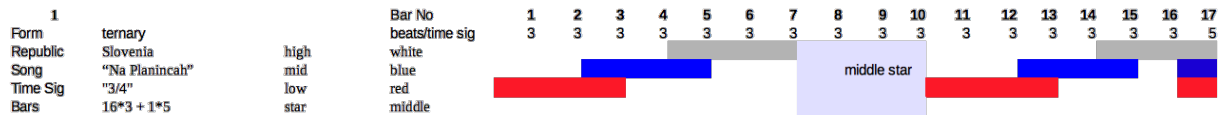


Figure 3.42: *Vexilla*, part 1 Structure

The time signature for part 1 of *Vexilla* was therefore set to 3/4, matching the time signature of the selected folk song. From the given length of the part (35.2 seconds) and the preset tempo (90 bpm), I calculated that there should be 53 beats in the part. The bar structure of the part was then constructed based on the calculated number of beats as follows:

$$16 \text{ bars} * \frac{3}{4} + 1 \text{ bar} * \frac{5}{4} = 17 \text{ bars}$$

This bar structure was then presented in tabular form, as shown in Figure 3.42. The visual features of the Slovenian flag (see Appendix C) were mapped into the structural tables (Figure 3.42), serving as metadata for generating musical content.

The common feature appearing on all flags used in section 1 is a red star. It was mapped to a recognisable musical gesture and repeated in all structural parts, albeit in slightly different forms. In part 1, the star gesture (Figure 3.44) is placed in the middle of the temporal structure

(Figure 3.42). On either side of the star gesture, the flag’s coloured stripes are laid out in vertical order and mapped to a sound category as described in Figure 3.40. This creates a ternary form that is also used in other parts with a similar flag layout.

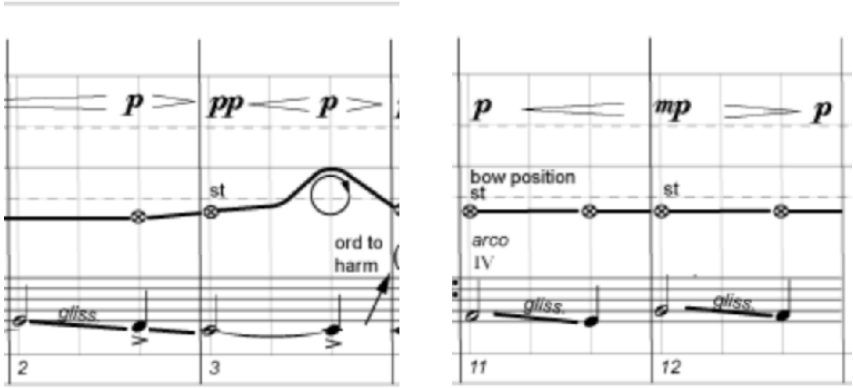


Figure 3.43: *Vexilla*, Cello part using *Na Planincah* melody elements

The “Na Planincah” melody served as inspiration for compositional material for all the instruments. Excerpts from the Cello part, as shown in Figure 3.43, are the most recognisable elements from the folk melody.

Other section parts were constructed following a similar process as described above. The tabular structure representation of parts in section 1 is shown in Figure 3.45 whilst the parts’ structures for sections 2 and 3 are displayed in Figure 3.46.

The figure displays three systems of musical notation for measures 8, 9, and 10, detailing various performance techniques and fingerings.

System 1 (Measures 8-10):

- Measure 8:** closed slap tongue, soft air, *p*.
- Measure 9:** air, *p*.
- Measure 10:** open slap tongue, air, *p*.

System 2 (Measures 8-10):

- Measure 8:** LH dampen strings, col legno ricochet + bow gliss, bow position, bow gliss., *p*.
- Measure 9:** air noise II, LH gliss., III, III, col legno battuto, *p*.
- Measure 10:** LH dampen strings, bow gliss., IV, col legno ricochet + bow gliss., *p*.

System 3 (Measures 8-10):

- Measure 8:** *p*, col legno battuto, IV, harm.
- Measure 9:** *p*, col legno bow vertical gliss., bow gliss., III, II, III, LH dampen strings, harm.
- Measure 10:** *p*, col legno battuto, dampen strings, harm.

Figure 3.44: *Vexilla*, part 1 star gesture

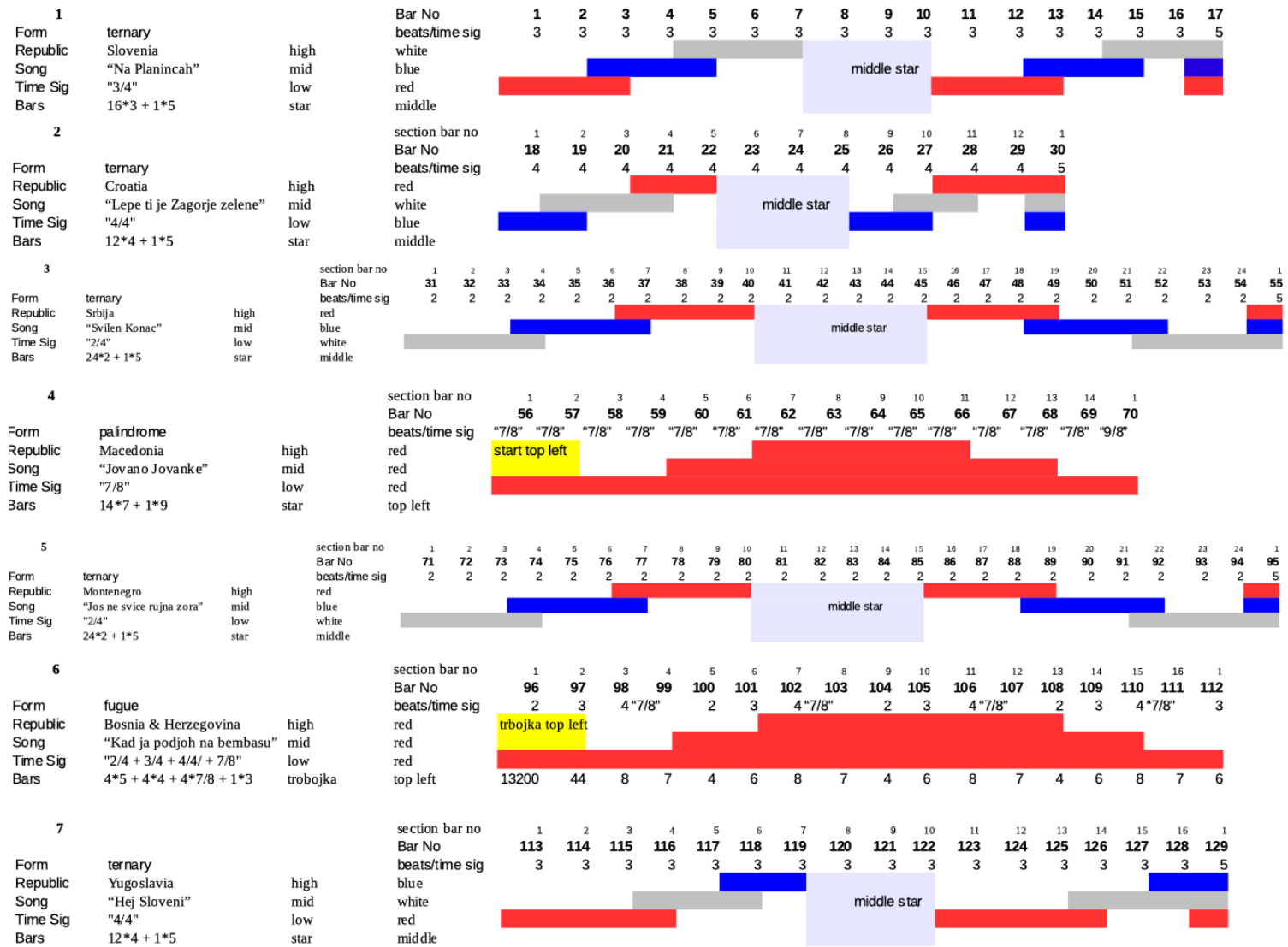


Figure 3.45: Vexilla, section 1 parts structure

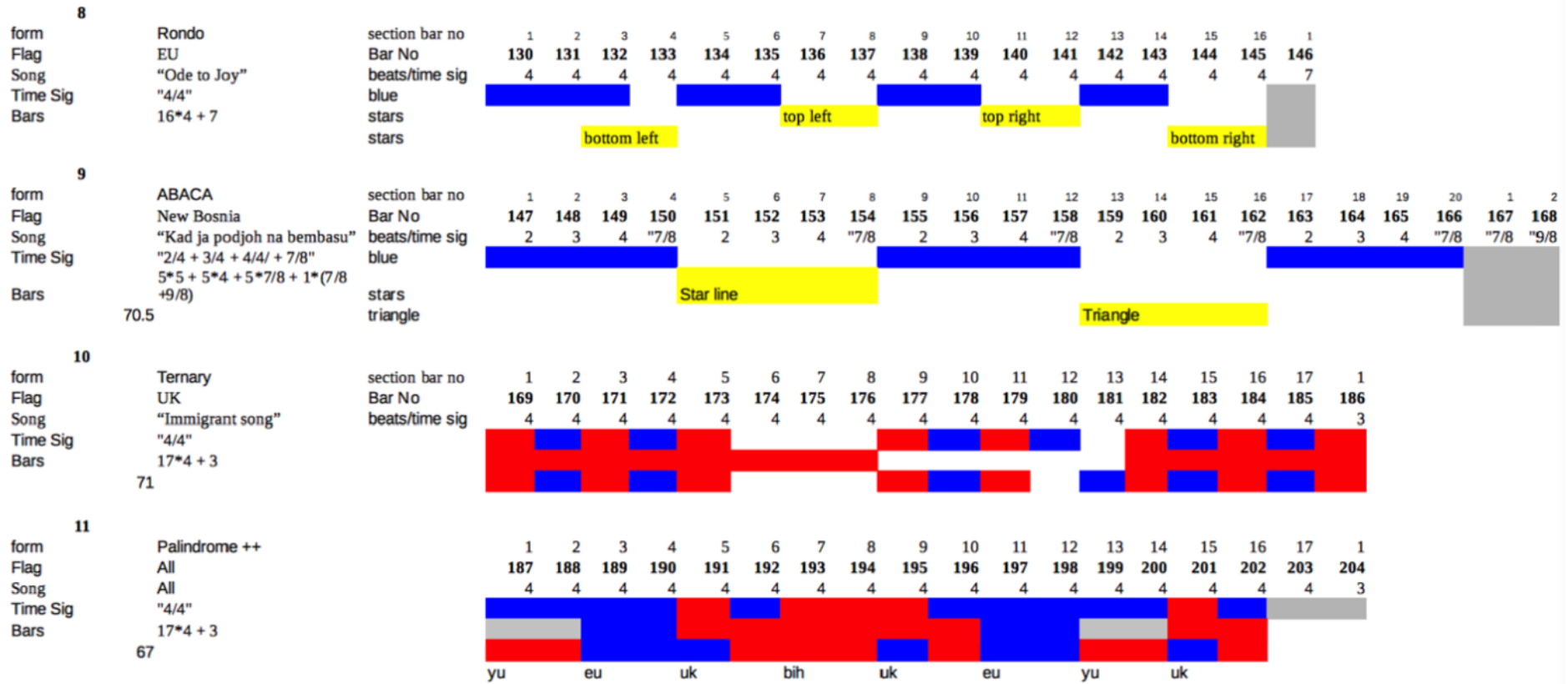


Figure 3.46: *Vexilla*, sections 2 and 3 parts structure

3.3.2.1 The Use Of Colour In Music Notation

The additional experimentation involved using colour in a graphic score, as shown in Figure 3.47. The aim was to associate a specific playing technique with a colour, thereby potentially increasing the player's score reading and comprehension speed compared to standard symbolic notation. For instance, Bass Clarinet multiphonics were associated with the colour blue. Whilst the idea received positive feedback, it was concluded that the use of colour should be employed sparingly and might be better suited for musical dynamics visualisation. A concern raised was that colour-blind musicians could be disadvantaged if this technique was extensively used and there was no other way to ascertain the meaning. In subsequent portfolio pieces, I ensured that the meaning of any coloured notation elements could be deduced from supplementary cues, such as their position on the staff or the direction of movement.

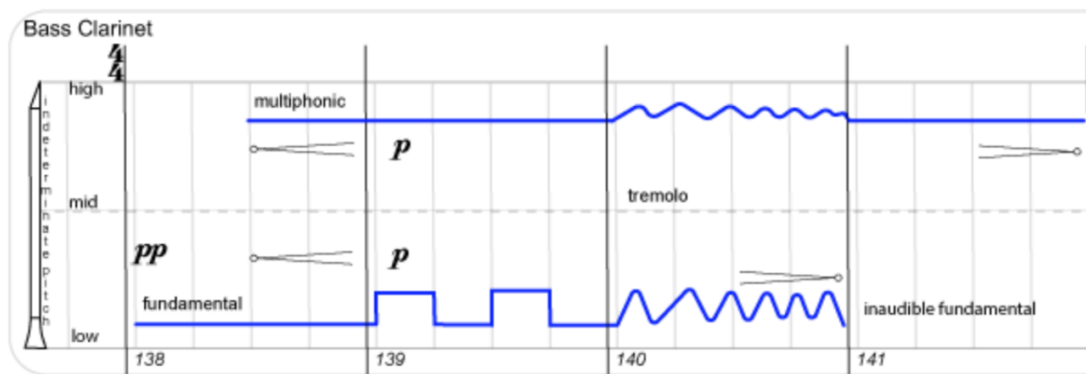


Figure 3.47: *Vexilla*, use of colour in the notation

3.4 Comprov

With:
String Quartet

Duration:	5 - 30 minutes
Links:	Full Score ZScore Package <i>Comprov</i> Exposition recording Comprovisation recording Notation overlays walkthrough video

Comprov notation and the required system implementation were developed for the workshop with the Ligeti Quartet in November 2019. The score material used during the workshop was derived from the first 8 pages written for *Union Rose* (3.5). *Comprov* play starts with the Exposition, where the first eight pages are played exactly as written. It then continues with dynamically generated pages based on the Exposition material in a continuous loop for as long as is deemed appropriate.

The primary objective of the workshop was to explore real-time, conductor-led networked comprovisation based on the unconstrained positioning of the dial between static pre-composed and dynamic notation, as illustrated in Figure 1.3. This was achieved with the novel concept of dynamic notation overlays specifically designed for comprovisation (3.4.1.2). The instrument stave layout was redesigned to visually separate performance parameters like bow speed, pressure, and position within a two-dimensional space (3.4.1.1). These areas could then be dynamically overridden by notation overlays controlled by a conductor. The ZScore Control GUI was enhanced with dynamic overlay controls (3.4.1.3) that enabled real-time movement of the dial, as illustrated in Figure 1.3. Additionally, the ZScore server design was improved to accommodate a generic Strategy interface, allowing for system behaviour modifications as required by a specific score. Two Strategies were implemented for *Comprov*: Randomisation (3.4.1.4) and Continuous Play (3.4.1.5). This approach allowed for a section of the composition material to be looped and drastically altered throughout an extended performance, resulting in a wide range of musical output.

3.4.1 Objective: Networked Comprovisation

3.4.1.1 Notation Segmentation

ZScore allows for syntax-independent and constraint-free positioning of notation elements, making it suitable for various dynamic notation styles. The design preference prioritises usability, aiming to offer a consistent layout familiar to classically trained musicians. Early versions of the notation layout employed a mixed symbolic/graphic notation space, which provided optimal and flexible screen real estate utilisation (Figure 3.24). However, after gathering feedback through research questionnaires and verbal conversations with workshop and performance participants, it

became evident that most musicians preferred the distinct and consistent positioning of different performance parameters rather than the mixed space approach. Moreover, certain notation element positioning felt more “natural” to performers, such as having the dynamic markings always displayed below the pitch information. Taking into account these considerations and the results from multiple user trials, the layout design illustrated in Figure 3.48 was established for *Comprov* sessions.

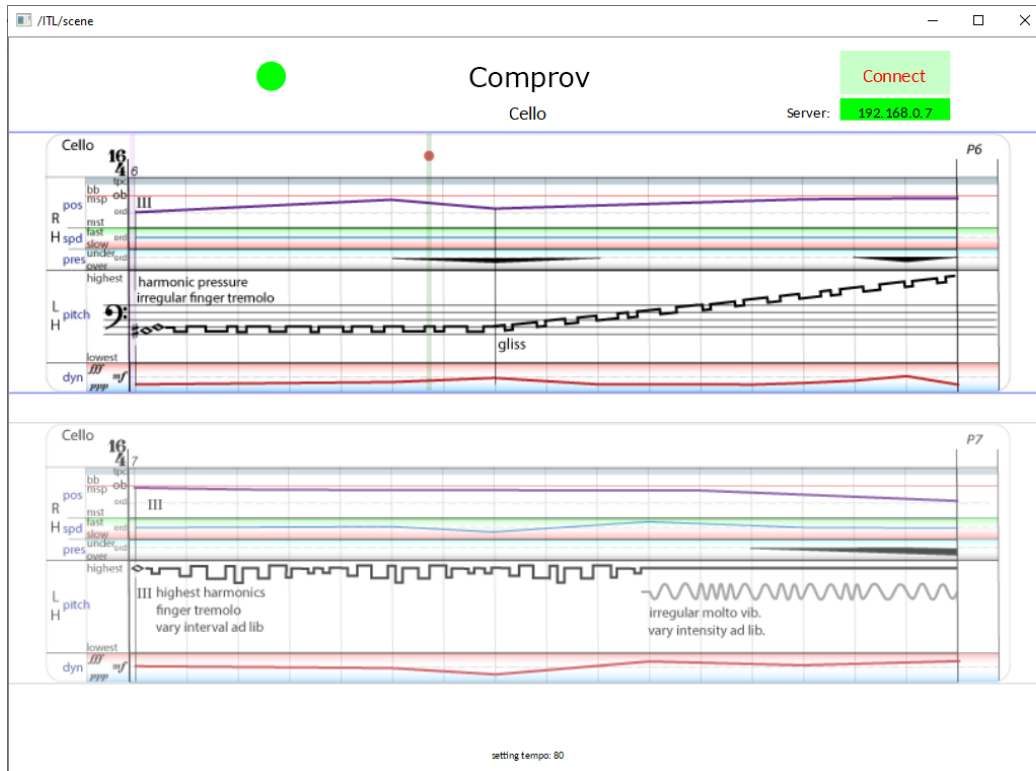


Figure 3.48: *Comprov*, string instrument notation layout

Performers intuitively apply complex playing techniques when reading a notated score, drawing from years of learning, practice, and performance experience. For instance, classically trained string instrument players automatically translate music dynamic markings into various bowing techniques, skillfully controlling the bow’s speed and pressure. String players strive to produce a sound quality that aligns with the aesthetic requirements of the specific tradition or style. This anticipation of the desired “ideal” sound leads to a learned application of the appropriate bowing techniques. In the latest pieces written for ZScore, the artistic aim is to create a particular sound quality by intentionally decoupling the embodied relationships between the notation, playing techniques, and sound through innovative dynamic notation techniques that allow for the flexible control of different playing techniques in real-time.

The user trials and various concerns described above led to the development of a notation layout where the key performance parameters are separated into distinct two-dimensional Cartesian coordinate spaces. Figure 3.49 illustrates a vertical staff layout designed for string instruments, taken from the score shown in Figure 3.48. The staff is vertically divided into

three main sections dedicated to the right hand, left hand, and music dynamic notation. The right hand notation section is split into three subsections: position, speed, and pressure notation. These subsections primarily refer to bowing techniques but can contain other actions such as pizzicato or percussive sound gestures.

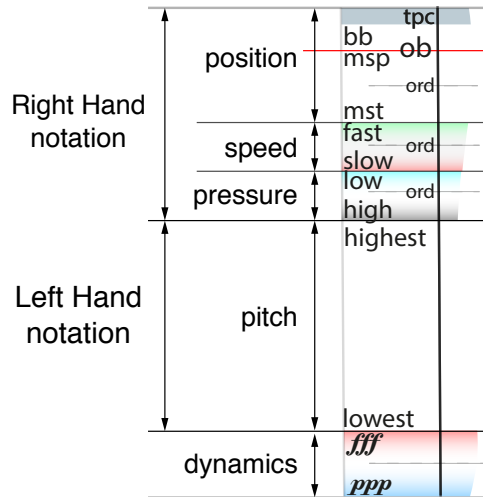


Figure 3.49: *Comprov*, vertical stave layout

The left hand notation contains pitch or playing technique information in either symbolic or graphic notation. The dynamics section is consistently displayed below the pitch information, based on feedback from musicians. In each section (Figure 3.49), the vertical axis represents a named performance parameter value, with a parameter-specific vertical range. For example, the bow position range for all string instruments starts with “molto sul tasto” (mst), then continues with markers for ordinary playing position (relative to the given dynamics), “molto sul pont” (msp), “on bridge” (ob), “behind bridge” (bb), and finally ends with the “tailpiece” (tp) marker. The horizontal axis represents time and remains identical for all performance parameters. The position cursor, rendered as the green line across the stave (Figure 3.50), shows the current position on each stave at any point of time.

Additionally, the dynamic beat cursor, represented as a red bouncing ball, indicates the onset of each beat, akin to a conducting gesture. The cello part excerpt (Figure 3.50) taken from the *Comprov* score (Figure 3.48) illustrates a pre-composed static notation layout. Various performance parameters, such as position, speed, and dynamics, are visualised as continuous lines, indicating a gradual change in their current values. Pressure notation employs coloured geometrical shapes indicating the amount of bow pressure that needs to be applied relative to the current dynamics. The left hand notation, in this case, combines textual, symbolic, and graphic elements to convey finger pressure, position, and timing information. The finger tremolo timing information serves as an approximation, illustrating the idea of irregular finger placement. Performers are given the freedom to use their aesthetic judgement to decide the exact tremolo timing during play. This decision might vary each time the section is performed, as the overall context and surrounding sound output can change with each pass, as described below.

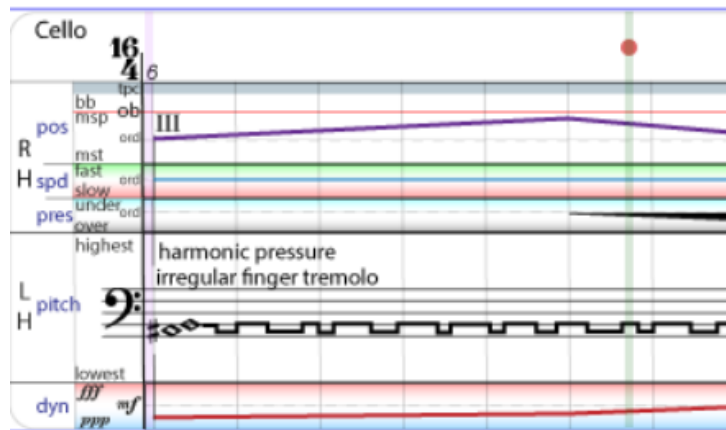


Figure 3.50: *Comprov*, Cello notation excerpt

3.4.1.2 Dynamic Notation Overlays

Real-time dynamic decision making and notation rendering require a user interface design that allows dynamic elements to update at any point of time without any detrimental impact on the displayed static notation elements and musicians' look-ahead preparation time. Dynamic notation updates need to be clear, easily understandable, and suitable for real-time cognitive processing. Therefore, complex notation updates should preferably be scheduled for display in predetermined time window slots, as described in Section 3.2.1.2. As a general guideline, the performance continuity should not suffer at any point during dynamic notation updates, unless it is an intentional side effect.

One of the reasons that led to the separation of performance parameter notation was the need to provide dynamic parameter value overrides, aimed at improvised music-making. To achieve dynamic overrides, each parameter was assigned a graphic overlay (Zagorac 2020). Overlays sit on top of the static notation covering the entire Cartesian space assigned to the corresponding performance parameter. The current dynamic parameter value is rendered either as a horizontal line on top of the overlay and/or as an overlay background colour value. Figure 3.51 illustrates the position, speed, pressure, and music dynamic overlays for the same notation excerpt shown in Figure 3.50.

In this case, only the pitch information remained the same as the precomposed static notation. As can be observed in Figure 3.51, each parameter's current value is represented by the coloured line (red for dynamics, purple for position, blue for speed, grey for pressure) and the background colour covering the entire two-dimensional space assigned to the parameter. In this instance, the player is asked to play forte sul pont with a fast bow and strong overpressure. Bow speed and position markings are always relative to the indicated music dynamics. Parameter values are controlled from the ZScore control GUI described below and have a preset range [min = 0, max = 100]. The parameter's line position is obtained by mapping the current parameter value to the vertical space assigned to the parameter (minY, maxY). Similarly, the background colour is interpolated from the range representing the minimum value (e.g. for dynamics it is blue [R=0,

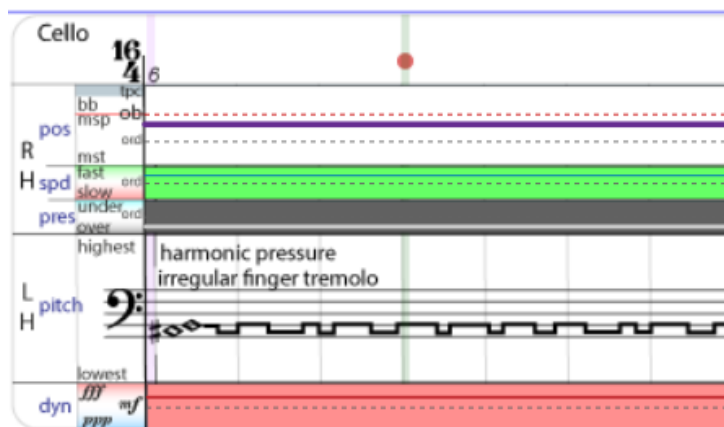


Figure 3.51: *Comprov*, dynamic notation overlays for position, speed, pressure, and dynamics

G=0, B=255]), the mid-value (white [R=255, G=255, B=255]), and the maximum value (red [R=255, G=0, B=0]). Overlays and parameter values can be set independently at any point in time and are immediately displayed on the musicians' screens.

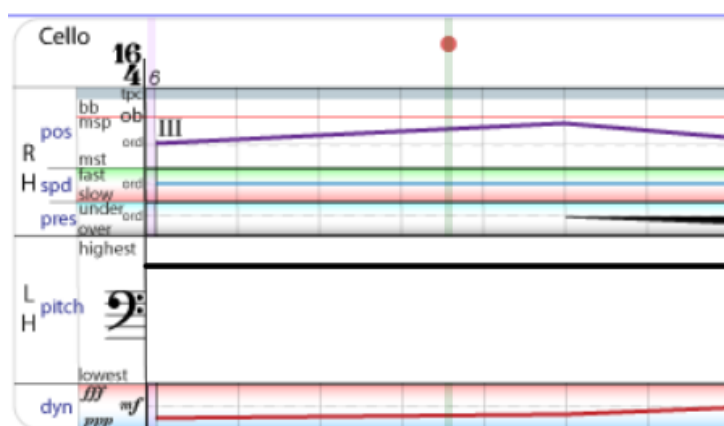


Figure 3.52: *Comprov*, dynamic pitch notation overlay

Figure 3.52 illustrates a different overlay configuration for the same score excerpt as shown in Figures 3.50 and 3.51. In this configuration, only the pitch parameter has been overridden with the overlay, indicating an approximate pitch to be played within the instruments range [min = lowest, max = highest possible pitch]. All other performance parameters in this example are precomposed. Any overlay can be switched on or off during a performance and their corresponding parameter values can be set independently at any point of time. Any parameter changes are immediately displayed on the musicians' screens.

3.4.1.3 Comprovisation Controls

ZScore's Control GUI has been extended to provide dynamic performance parameter controls (Figure 3.53). Performance parameter overlays described in the previous chapter can be switched

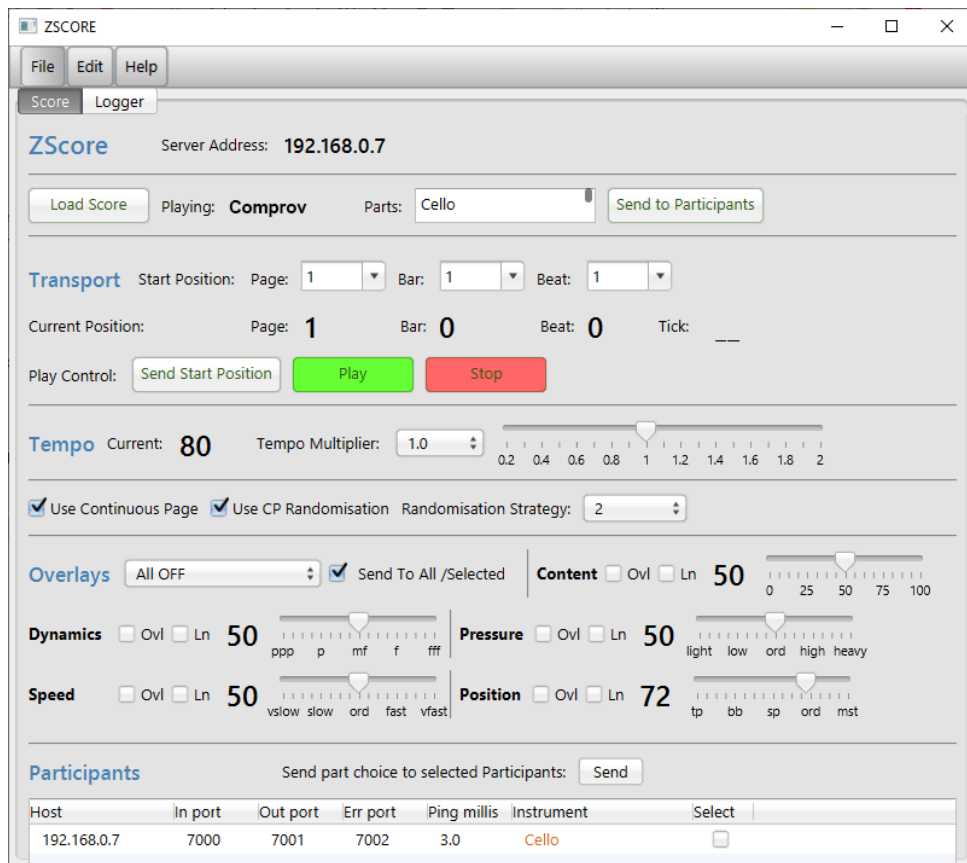


Figure 3.53: *Comprov*, Control GUI

on or off at any point during a performance from the Control GUI. Each overlay (e.g. Dynamics, Speed, Pressure, Position, and Content) is assigned a set of controls that govern its appearance. These controls include checkboxes that enable the overlay and related line indicators, as well as a slider that sets the overlay performance parameter value.

The performance parameter ranges are displayed in musical terms. For example, the music dynamics range is displayed as “ppp - p - mf - f - fff”. The value selected by the performance parameter slider is scaled to the related overlay line position and background colour on musician’s views, as explained above.

The Control GUI also contains a number of presets that set various combinations of overlay layouts. The default preset is “All OFF”, which disables all overlays. Other presets switch particular overlays on or off. For example, the “All ON Content OFF” preset enables all overlays apart from the pitch overlay. It is also possible to use overlay background colours only without the line indicators by selecting the “All ON Lines OFF” preset.

The administration GUI communicates with other performance participants through ZScore’s server, which provides notation scheduling and a distribution service. The server also receives events sent by other performance participants, processes them through an algorithm that analyses incoming events, and validates decision logic. The outcome of the decision logic is then passed to the score processing and scheduling engine, which identifies the required notation and distributes

it back to the performance participants. Unlike other scheduled events, overlay control messages are passed through to the musicians' clients immediately.

3.4.1.4 Randomisation Strategy

The composed music material in *Comprov* is limited and continually reused, as explained in the introduction (3.4). The Score Randomisation strategy was developed to provide a mechanism for this reuse of material that is suitable for a improvised performance with a string quartet. The Score Randomisation algorithm determines the notation and instrumentation to be used in the next time window calculated by the scheduling engine. Figure 3.54 displays the available Randomisation strategy configurations in *Comprov*, designed for a string quartet.

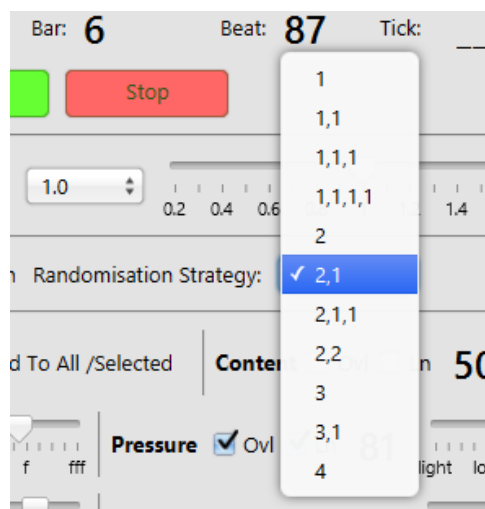


Figure 3.54: Randomisation strategy configuration control

The configuration determines the number of distinct score pages and instruments that should be used. The term “page” is equivalent to the notation displayed in a single pane. For example, the configuration value “2” means that two randomly selected instruments should play the same randomly selected page; the configuration value “2,1” means that two instruments should play the same page, and one instrument should play another randomly selected page; “1,1,1,1” means that all four string quartet instruments should play a randomly selected page, and so on.

Comprov pages have been constructed to function as an independent unit, reusable in various vertical structure combinations. The aesthetic impact of sounding page combinations is evaluated in real-time during a performance. Any undesired sound elements can be modified with dynamic performance parameter controls to rebalance the page combination’s musical outcome.

3.4.1.5 Continuous Play

Comprov starts with the fully notated Exposition, and then transitions into a improvisation based on the Score Randomisation strategy settings and manual Overlay controls. To facilitate uninterrupted page scheduling on the server-side, I introduced the concept of a “Continuous

Page”. When the “Continuous Page” setting is enabled on the Control GUI (Figure 3.53), the server automatically creates and schedules a new page when the last scheduled page is reached during a play. The content of the “Continuous Page” is determined by the Score Randomisation strategy settings (3.4.1.4) for each instrument at the time of page creation. This continuous play can run indefinitely until the Stop button on the Control GUI is pressed.

3.5 Union Rose

For:
String Quartet
Audience
and Max

Duration:	10 minutes
Links:	Full Score ZScore Package Score follow video Workshop video recording Setup instructions video

3.5.1 Objective: Interactive Composition And Performance

Union Rose represents a major step forward in terms of interactive networked performance features, the scale of the composition challenge, and the technical implementation of the ZScore system. The piece takes inspiration from the magnificent east window rose (Figure 3.55) at the Union Chapel in London, where it was originally intended to premiere in 2020. Each composer involved in the event selected a feature from the venue as inspiration for their compositions. Unfortunately, due to restrictions during the COVID pandemic, the planned event never took place as scheduled. *Union Rose* eventually had its first performance by the Ligeti Quartet in February 2022.

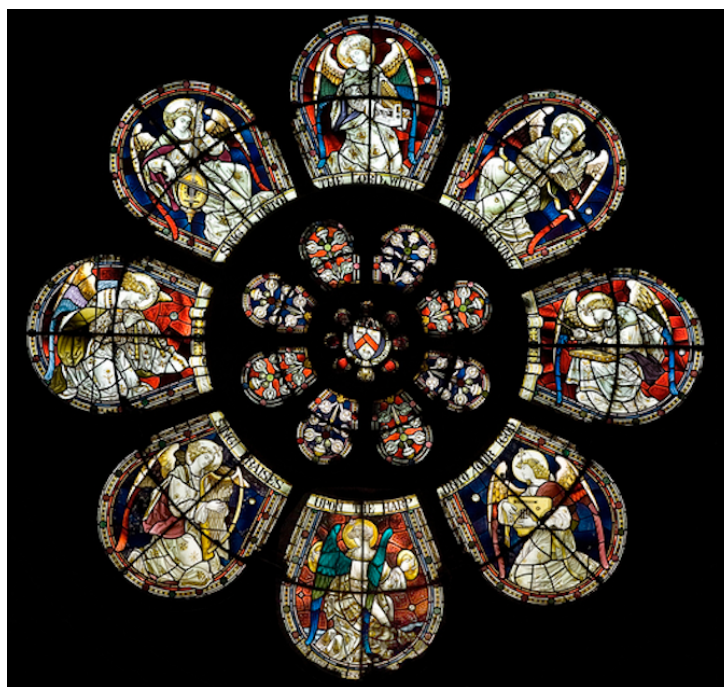


Figure 3.55: Union Chapel Rose⁶

The ZScore system was enhanced to allow for audience participation in performances. Audience members could access an interactive score representation on their mobile device

⁶Source: <https://unionchapel.org.uk/projects/heritage/roses-of-union-chapel> (last visited 3 Mar 2020)

web browser, allowing for participation in decision-making and sound production (3.5.1.2). For example, the audience could choose what music material was played next in section A' by selecting the corresponding tile associated with the desired page of music (Figure 3.77). Furthermore, they could trigger digital audio and speech patterns in sections B' (Figure 3.84) and C' (Figure 3.89).

The musicians' notation was moved to a web container that can run on any digital device, replacing INScore standalone client (3.5.1.1). The notation segmentation was further enhanced to provide a cleaner visual layout with interactive features that also allowed the musicians to participate in the decision-making process. For example, the musicians could choose what instrumentation was used in the next page of music in sections A', B', and C', as illustrated in Figure 3.57 and explained in the score commentary for these sections.

A networked digital audio node was developed for the Max platform, comprising an external object for network communications, a jsui visual interface for score information, and a set of custom patches for audio processing (3.5.1.3). These ZScore Max patches could be controlled in real-time either through network events triggered by the score or directly through MIDI controllers or a computer keyboard/mouse. The score and server were enhanced to enable the integration of new components into a coherent, synchronised composition model. Consequently, the scope of the composition process expanded significantly, encompassing audience score visualisation, an interactive score design, and digital audio sound production. These innovations represented crucial steps in the development and refinement of the ZScore system.

3.5.1.1 Web ZScore

A desire to ensure wider compatibility with as many modern mobile devices as possible led to the porting of the INScore notation view into a standard web browser implementation. To achieve this, several custom JavaScript libraries were created to provide the required functionality, including connectivity, dynamic score visualisation, interactivity, and synchronisation features. The Web ZScore JavaScript libraries and their brief descriptions are shown in Figure 3.56.

The only third-party JavaScript library used in Web ZScore is the GreenSock GSAP library (2014), which was utilised for the position indicator animation.

Dedicated HTML and JavaScript files were created for each score containing details and functionality specific to the particular composition, whilst common functionality was extracted into the libraries listed in Figure 3.56. When the musician's browser connects to the server, the initial view displays a list of available scores. This list can be modified for each performance as required. Musicians can return to this list by clicking on the "home" icon in their part views.

In general, all score parts have an identical alternating pane layout, as described in Section 3.2.1.5. An example of the Web ZScore Cello part notation is shown in Figure 3.57.

The notation look and feel is identical to the INScore version (Figure 3.48), albeit with several enhancements. The current tempo is now displayed in the top left corner, just below the traffic lights. As the tempo can be manually modified, it is useful to always see the current value,

library	description
zsNet.js	Websocket and SSE connectivity
zsSvg.js	SVG drawing and helper functions
zsMusic.js	Pitch, transposition, rhythm etc. definitions and functions
zsUtil.js	Various utility objects and functions
zsWsAudio.js	Web score audio, metronome (uses zsBeep.js)

Figure 3.56: Web ZScore JavaScript libraries

The screenshot displays the Web ZScore interface for the cello part of 'Union Rose'. At the top, there are four circles (three white, one green) and a metronome icon with a tempo of 80. The title 'Union Rose' is centered, with 'Cello' below it. A green 'READY' button and a home icon are in the top right. The score is divided into two systems, P5 and P4. System P5 includes instructions like 'vertical bow tremolo', 'behind bridge', 'I,II,III,IV ord arpeggio', 'arco ord III', and 'lightly mute strings'. System P4 includes 'arco ord', 'pizz', 'arco', 'III vertical bow tremolo', 'II,III', 'irregular finger battuto', and 'mute strings'. A control bar at the bottom has buttons for 'Cello' and 'Violin1', with two additional greyed-out buttons.

Figure 3.57: *Union Rose*, cello part in Web ZScore

which may deviate from the value shown in the score. An audible beep tempo indicator can be activated by clicking the metronome icon located to the left of the tempo value. The tempo beep frequency can be adjusted for each part for easier identification. This device is mainly used for synchronisation with Max digital audio. The home button, located in the top right corner, opens the main score menu page.

The most notable change is the addition of the action bar at the bottom of the Web ZScore notation view. In *Union Rose*, the action bar displays buttons that allow musicians to interactively opt in or out of participating in the next page. In Figure 3.57, for example, the next page is

scheduled to be played by Cello and Violin 1. The Cello button is highlighted in light green to enable the player to easily determine whether their part is involved in the next page (the example view is for the cello part). A player can click on the Cello button to opt out of participating in the instrumentation scheduled for the next page. If the cello part was not scheduled to play the next page, the cellist could choose to opt in by clicking on any available instrument to take their place in the next page's instrumentation. The time window when buttons are visible is managed by the server based on the score configuration. Buttons can be used as long as the time window is open. This can create a game-like race between players during the time window, as the last person to click the button wins.

The INScore time-space synchronisation map was reused in Web ZScore for position tracking purposes. During the page initialization process, a GSAP timeline is created for each stave. This timeline consists of several GSAP tweens responsible for animating position tracking. A GSAP tween is a lightweight JavaScript object that controls animations by gradually transitioning the values of an element's properties over a specified duration. Tweens are constructed from the data available in the time-space beat map sent by the server, which includes spatial x and y coordinates for the start and end positions of each beat in the stave, as well as the duration of each beat. These GSAP tweens govern the movement of both the vertical position tracker and the beat ball for each beat. Individual beat tweens are then added to the corresponding stave's GSAP timeline. Furthermore, a label marking the onset of each beat is inserted into the timeline. This beat onset label allows for setting the starting position to any desired beat within the stave.

All graphical files required for the part are preloaded by the web client on score load (or section load in larger scores), thus reducing any potential latency and jitter issues during a performance.

3.5.1.2 Audience Score Representation

A web audience score view was created to allow any Wi-Fi enabled mobile device running any Internet browser brand to connect and participate in a performance. Most of the JavaScript libraries developed for the Web ZScore notation (Figure 3.56) were reused in the audience view. Additionally, a set of libraries dedicated to audio generation were developed using Web Audio API (2011) and Web Speech API (2012) (Figure 3.58). Similarly to the Web ZScore notation front end, all animation tasks within the audience view are implemented with the help of the GreenSock GSAP third-party library.

In *Union Rose*, the design of the audience score representation is based on the Union Chapel rose (Figure 3.55). A photo of the rose was converted to SVG format with a reduced colour palette and stylised in Adobe Illustrator (Figure 3.59). The essence of the rose's geometry and the content of its shapes were preserved, whilst the religious symbolism was transformed into more abstract forms.

One example of the shape conversion is the replacement of the religious halo and the sacred human heads in the outer circle with abstract shapes representing two cerebral hemispheres

library	description
zsAudio.js	Parent library, handles audio tasks, built on top of Web Audio API requires libraries below
zsGranulator.js	Granulator implementation
zsSpeech.js	Handles device's speech, uses Web Speech API
zsPlayer.js	Buffer player implementation
zsNoise.js	Noise generator
zsSynth.js	Simple synthesizer implementation

Figure 3.58: ZScore JavaScript audio libraries



Figure 3.59: *Union Rose*, stylised SVG rose for the audience view

(Figure 3.60).

Concentric circles cutting through the rose shapes were emphasised to underline the ideas behind the score, as explained below. Prompted by the obvious geometrical structure, the SVG rose was split into three main groups: “centreShape”, “innerCircle”, and “outerCircle”, each consisting of a number of SVG paths (Figure 3.61).

The entire SVG rose was then covered with a concentric circle grid structure of SVG paths resembling a vitrage of tiles that also acts as a window, gradually revealing the rose (Figure 3.62). The reference to a tiled window also reflects the original function of the Union Chapel rose.



Figure 3.60: *Union Rose*, one of the stylised SVG shapes

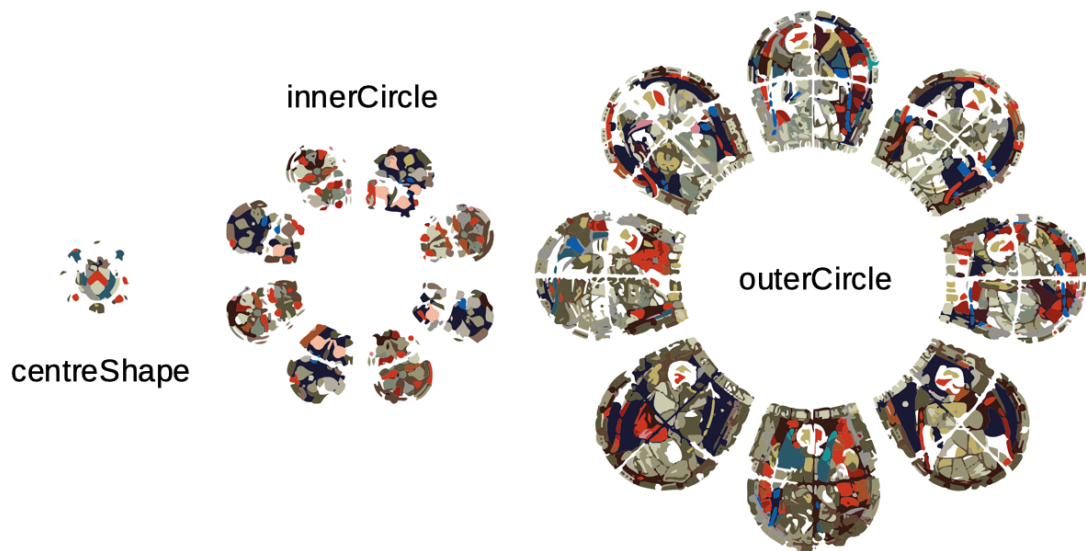


Figure 3.61: *Union Rose*, shape groups

As each concentric circle of the rose contains eight shapes, a decision was made to create an 8 x 8 grid consisting of 64 tiles. The idea from the start was to map tiles to score pages, and therefore, a score containing 64 pages was created in Adobe Illustrator. “Tile” objects were created both on the client and server-side, containing properties such as text value, click count, page mapping, style, etc. A mechanism maintaining “Tile” state synchronisation between the client and the server-side was implemented, enabling visualisation management directly from the score.

In order to reduce the amount of information required to maintain the state of the visualisations in the score, a configuration file `audienceScoreConfig.yml` (written in YAML (2004) format) was added to the build. The audience score configuration file contains a number of presets that can be invoked from the score by using a preset id. Presets can contain configuration

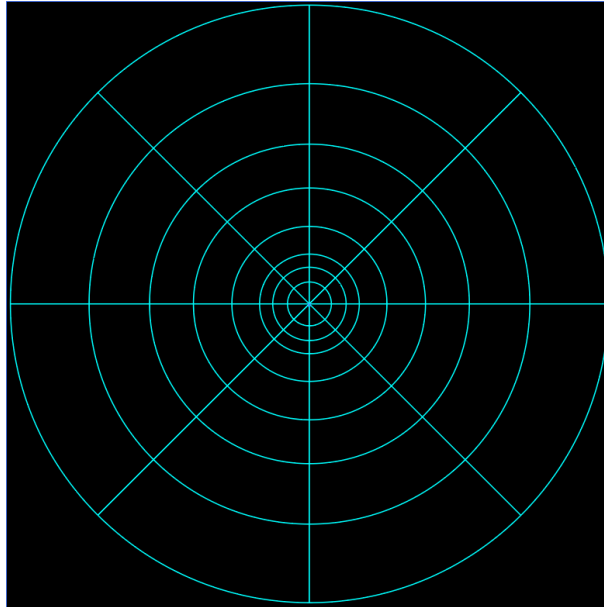


Figure 3.62: *Union Rose*, SVG rose cover grid consisting of 64 tiles

values or scripts as illustrated below:

```
presets:
  - id: 2
    config:
      - granulator: {
          masterGainVal: 0.1,
          maxGrains: 16
        }
  - id: 3
    scripts: [
      "webScore.setZoomLevel('outerCircle');",
      "webScore.setVisible(['centreShape'], true);",
      "webScore.setVisibleRows([1, 2, 3]);",
      "webScore.setActiveRows([1]);",
    ]
```

Additionally, the audience score configuration contains the “Tile” to score Page mapping as follows:

```
pageRangeMapping:
  - tileRow: 1
    tileCols: { start: 1, end: 8 }
    pageRanges: [ { start: 1, end: 8 } ]
    assignmentType: SEQ
```

Tiles are identified by their position in the grid, referenced by their row and column indexes. In the example above, eight tiles in the first row are mapped sequentially to the first eight pages of

music. The assignmentType can be SEQ (sequential) or RND (random). When SEQ assignment is used, the tiles are mapped in ascending order (column 1 → page 1, column 2 → page 2, etc.), whilst with RND assignment, the tiles are mapped randomly within the range (e.g. column 1 → page 4, column 2 → page 1, etc.).

The configuration file is loaded on the server-side. When the score invokes a configuration preset id, the content of the preset is processed and sent to clients. The ZScore Java server uses the Nashorn engine (2012) to interpret JavaScript commands and execute appropriate methods.

3.5.1.3 Digital Audio

ZScore can communicate with any digital device supporting the OSC protocol over a local network. For digital audio tasks, the powerful Max (1990) programming environment by Cycling '74 was initially selected, primarily because of its support for networking and various modes of programming. To establish communication between Max and ZScore, a mxj (Max Java) object named “zscore” was created to act as a client to the ZScore server. Furthermore, a Max jsui (JavaScript User Interface) object containing zsui.js was developed to display score information in the Max graphical user interface (Figure 3.63).

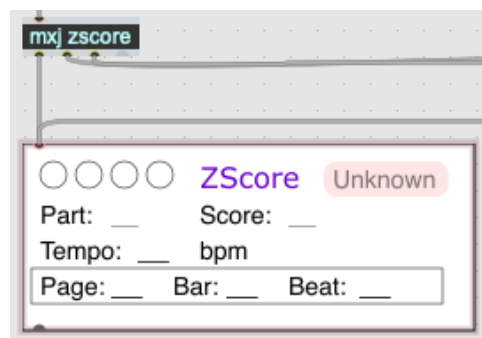


Figure 3.63: ZScore Max mxj and jsui objects

The mxj “zscore” object uses a proprietary network communication layer, utilising solutions similar to those applied on the server-side. It receives and sends UDP messages that are passed to internal Max objects via input and output Disruptors (2011). For client compatibility, it also implements a connection handshake method similar to INScore.

The “zscore” mxj object has three outlets. The first outlet sends messages to the jsui object, containing score information and low-priority commands. The second outlet sends high-priority commands directly to other components. Currently, only the “play” and “stop” commands are defined as high-priority for more accurate synchronisation with acoustic instruments. The third outlet sends a bang signal on each beat to the Max “Click” bpatcher, which produces an audio beep used for tempo synchronisation with the musicians’ front ends.

A set of objects implemented as bpatchers were then added to the main Max patch, providing digital audio playback and processing. These include a multichannel groove processor (ZS MC Groove) built around the native mc.groove~ object, custom-made granular synthesis (ZS Gran-



Figure 3.64: ZScore Max patch

ulator), and several buffer payback objects (BufGroove 1-4). All audio objects can send signals to several effects, including a tapin~/tapout~ delay, a reverb based on the gen~ implementation of Gigaverb by Olaf Mathes, and a simple biquad~ filter. Most of the object parameters can be controlled directly from the score via a network.

For convenience, a set of Max presets governing the patch state was defined for each composition. These presets can be recalled remotely by the server-side scripts or from the score. A screenshot of the main ZScore Max patch is displayed in Figure 3.64.

3.5.1.4 Score Scripting Extensions

As a means of providing automation for audience visualisation and digital audio processing, several new resource types were added to the embedded score scripting: “web” for audience visualisation scripts, “max” for digital audio scripts, and “sce” for the server scripting engine scripts. Figure 3.65, taken from the *Union Rose* score AV part, bar 25, illustrates the utilisation of the new scripting resource types.

The first line in the top left corner contains:

```
web:beat=-1:reset=only:webScore.reset(5);
```

The colon (:) is used as a delimiter separating different sections of the script. The identifier “web” indicates that this script is related to the audience score representation. The name-value pair “beat = -1” means that the script should be executed on the upbeat leading to bar 25. In embedded scripts, beat numbering refers to the beat count within the bar, whilst the minimum resolution for embedded script scheduling in *Union Rose* is one beat.

AV	16 4 25	J=120	P25
		<pre> web.beat=-1:reset=only:webScore.reset(5); web.beat=-1:reset=both:webScore.setSpeechSynthConfigParam('volume',1.0); web.beat=3:webScore.setInstructions('Turn up the volume!', 'Click your favourite tile', 'to hear it speak'); web.beat=1:webScore.setActiveRows([5], false);webScore.deactivateRows([1,2,3,4]); web.beat=1:webScore.setTitleTexts(['15-1','15-2','15-3','15-4','15-5','15-6','15-7','15-8'],['conspiracy','equality','monarchy','global warming','free market','sovereignty','europeanism','democracy']); web.beat=1:webScore.setAction('resume' 'TIMELINE' ['innerCircle']) ('duration': 64); web.beat=1:webScore.enableSpeechSynth(); web.beat=1:webScore.setZoomLevel('innerCircle'); web.beat=2:webScore.setAction('start', 'ROTATE', ['ctg5',ctg6]); web.beat=2:webScore.playTiles(['15-1']); </pre>	<pre> max.beat=-1:reset=only:setFile,b1,UnionRose_b17.wav max.beat=-1:reset=only:preset,8 max.beat=5:play.granulatorCont max.beat=9:play.granulatorContStop max.beat=8:stop.b2 max.beat=10:preset,8 </pre>

Figure 3.65: *Union Rose*, new scripting resource types

The next name-value pair “reset=only” indicates that this script should be executed only if the score play start is set to bar 25 from the performance Control GUI. A random position start requires the score state reset on both the client and server so that the play can continue with the relevant content. On the other hand, the value “reset=both” means that the script should be executed both on the score state reset and when the regular continuous play reaches the specified beat. The last part of the script “webScore.reset(5)” is a string literal evaluated by the scripting engine. In this case, the server executes the content of preset 5, which is defined in audienceScoreConfig.yml as:

```

- id: 5
  scripts: [
    "webScore.reset(1);",
    "webScore.setZoomLevel('innerCircle');",
    "webScore.setVisible(['centreShape'], true);",
    "webScore.setVisible(['innerCircle'], true);",
    "webScore.setVisible(['outerCircle'], false);",
    "webScore.deactivateRows([1,2,3,4]);",
    "webScore.setVisibleRows([5, 6, 7, 8]);",
    "webScore.setActiveRows([5]);",
    "webScore.setStageAlpha(0.0, 0.5);",
  ]

```

Preset scripts can execute other presets (e.g. the line “webScore.reset(1);” executes preset 1), thereby forming chained preset execution. This technique significantly reduces the amount of scripting in the score and improves script reusability, especially when certain actions are frequently required. The reference “webScore”, in this case, refers to the current audience score instance.

The script below can be found in the top right corner of the AV part, bar 25 (Figure 3.65):

```
max:beat=-1:reset=only:setFile,b1,UnionRose_b17.wav
```

Here, the “max” prefix indicates that the script is related to the Max digital audio client. *Union Rose* scripting supports only a single Max client, however, this could be expanded by

assigning different ids to each client. The Max script content is comma delimited. The first string (e.g. “setFile”) is the command name sent as an argument in the OSC message. The second string (“b1”) is the target identifier (BufGroove player 1). The target string is also used to construct OSC addresses. In this case, the OSC address is “/zs/b1” where “/zs” is the root of the ZScore patch OSC address space. The last string (“UnionRose_b17.wav”) is the name of the file to be loaded into the buffer. It is sent as an OSC message argument. When the OSC message is received in the Max patch, the setFile command is sent to the jsui object as it is not a high priority command. The logic implemented in zsui.js looks up a Max object named “b1” and sets the file name value from the OSC message arguments. For this command to work, the file (e.g. “UnionRose_b17.wav”) must be present on the machine running the Max patch.

The score script can call Max presets using a similar mechanism. The script in the bottom right corner of the AV part, bar 25, is as follows (Figure 3.65):

```
max:beat=10:preset,8
```

This embedded script sets preset 8 defined in the Max patch on beat 10 of bar 25.

After working on the *Union Rose* score for a while, it became apparent that there was a need for a general scripting engine capable of managing server-side behaviour directly from the score. Therefore, a dedicated score scripting engine was created to accept and process any score script of the “sce” resource type. An example in Figure 3.65 contains one of these scripts:

```
sce:beat=12:sce.setRndStrategy([1])
```

This script manages Randomisation strategy settings as shown in Figure 3.54. Instead of using the Control GUI value, the Randomisation strategy is set directly from the score to “1”, meaning that the next page will be randomly selected and played by one instrument only. The instrument is ultimately selected by the musicians, as shown in Figure 3.57.

3.5.1.5 Score State Management

As the work on audience visualisation and digital audio processing progressed it became obvious that there was a need for a score state reset mechanism whenever the score play was stopped and restarted at an arbitrary position. Scripting and interactive features introduced continuous changes on both server and client-side during the score play, making it impossible to determine the exact state of all nodes at any particular point in time. To address this issue, a concept of the “score reset point” was invented to satisfy the ZScore requirement of enabling stop and start at an arbitrary position. The intention behind a reset point is to set the server and all connected clients to an appropriate state relevant to the time of the reset point execution.

The timing of the reset points is associated with a beat onset, similar to embedded scripting. Reset points are required for any beat where there is a material change in audio-visual state, such as the beginning of a new section. If the play start beat does not contain any reset points,

then the nearest reset point going back in time will be loaded and executed when the new start position is set. It is important to note that reset points are specific to the scripting resource type (e.g. “web”, “max”, “sce”, etc.). For example, if a starting beat contains a “web” reset but not a “max” one, then the beat’s “web” reset and the nearest previous “max” reset will be loaded and executed to ensure that both scripting type states are appropriately set.

All embedded scripts and configuration files containing name-value pairs “reset=only” or “reset=both” are treated as reset points. When the reset value is “only”, the corresponding scripts are executed only when the manual score start is set to the reset point beat. If the reset value is “both”, then the corresponding scripts are executed both when the score is restarted manually and when the regular continuous score play reaches the reset point beat.

3.5.2 Score Commentary

Union Rose was inspired by the large rose in the Union Chapel in London (Figure 3.55). Whilst the Union Chapel is widely recognized as a music venue, it still retains its original purpose as a church on certain days, guided by the founding principle of “Friend for All”. The rose window was initially designed with a religious context in mind, intending to showcase its features highlighted by the natural light coming from outside. During music events hosted at the venue, the rose window transforms into a magnificent stage backdrop, often illuminated by stage lighting from within the hall. The dual context of lighting, both natural and artificial, along with the rose’s symmetrical structure, became the foundation for further compositional development.

In the centre of the rose (Figure 3.55) lies a heart-shaped emblem, surrounded by three concentric circles. Each circle contains eight shapes symmetrically positioned along its circumference. These shapes gradually increase in size going outwards from the centre of the circles, forming an onion-like layered structure. This design triggered an enquiry into the relationship between my personal perception of the rose and the context it is meant to represent. Having set aside purely religious connotations, I delved deeper into research on the layers of core human beliefs and the formation of a subconscious identity.

	Section	bpm	Type	Comment	Rings	Content range	Start Page	End Page
I. AM.	A	80	Composed	Ligeti 2 nd Quartet, Allegro Nervoso, bar 19-22	1	1 – 8	1	8
	A'		Audience+Players	Audience material select	2,3	1 – 8	9	16
I Believe	B	120	Composed	"Love Supreme" theme, Coltrane	4	9 – 16	17	24
	B'		RND+Players	Audience Mobile Audio	5	9 – 16	25	32
I Want	C	100	Composed	"Money Money Money", ABBA	6	17 – 24	33	40
	C'		RND+Players	Audience Mobile Audio	7, 8	1 – 8, 9 – 16, 17 – 24	41	48
Coda	A''	80	Composed	Coda, recap section 1, audience selection recap	0	1 – 8	49	56

Figure 3.66: *Union Rose*, high level structure

Based on the SVG rose’s division into three sections, as shown in Figure 3.61, a compound

trio composition form was defined as presented in Figure 3.66. The three primary sections were given the names: *I. AM.*, *I Believe*, and *I Want*, symbolising layers of human identity expanding from the innermost to the outermost, akin to the rose’s concentric circles. Additionally, *Coda* was incorporated as a recapitulation device, allowing for the repetition of the initial material in a modified form. All of the main sections follow a binary form, where the musical material is first played as written (e.g. A) and then repeated in a transformed shape (e.g. A’). The “Type” column in Figure 3.66 indicates the nature of the transformation, which will be further discussed in the following text.

The *Union Rose* tempo curve (Figure 3.67) reflects the circular nature of the rose by ending on the same tempo (80 bpm) as the first section. The “Comment” column (Figure 3.66) provides additional information about the inspiration behind each section’s musical material and the intentions for the repeated sections. The audience score view contains eight concentric rings of tiles numbered from 1 to 8, with 1 being the closest ring around the centre and 8 being the furthest (Figure 3.62). The “Rings” column (Figure 3.66) indicates which rings of tiles are active in a particular section. As there are eight shapes in the rose’s concentric circles, each section comprises eight pages of musical material. The “Content range” column (Figure 3.66) indicates the source material page numbers, whilst the “Start Page” and “End Page” columns show the elapsed page numbers for each section.

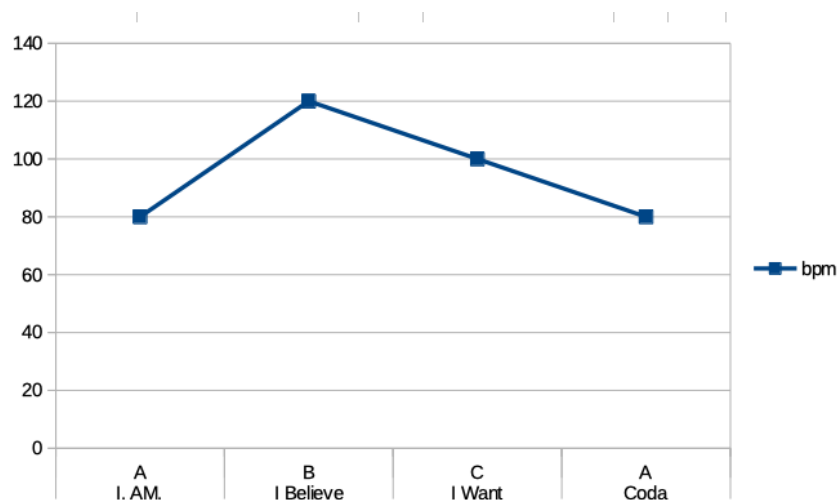


Figure 3.67: *Union Rose*, tempo curve

3.5.2.1 I. AM.

The section representing the innermost circle of the rose is titled *I. AM.*, as an homage to James Joyce’s *Ulysses*. In the episode titled “Nausicaa”, Bloom, one of the main protagonists, begins to write a message in the sand: “I. AM. A.”. But he runs out of space, leaving the sentence completion open to interpretation. Likewise, in *Union Rose*, an audience can complete the composition by selecting a word of their own choosing on their mobile devices.

The initial idea behind *Union Rose* was to write a piece for the Ligeti Quartet, drawing inspiration from the architectural features of the Union Chapel. However, I took it a step further by using the quartet's name as inspiration for the musical material. The pre-composition process included an analysis of György Ligeti's *String Quartet No. 2.*, specifically bars 19-22 of the first movement *Allegro nervoso*, which feature Ligeti's characteristic harmonic clusters that evoke a certain primordial quality. Figure 3.68 illustrates these harmonic structures with simplified rhythmic placements.

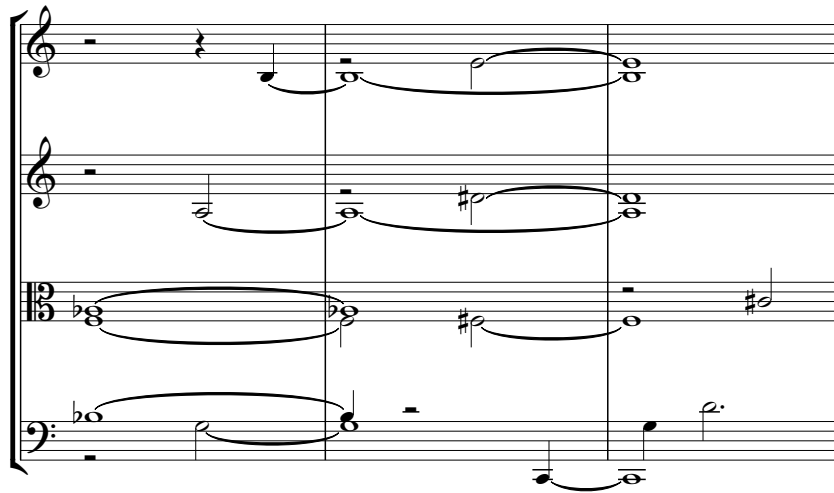


Figure 3.68: Ligeti, *2nd String Quartet*, bars 19-22 harmonic structure

In the next step, pitch sets for each section of *Union Rose* were defined from the harmonic structures in Figure 3.68. Initially, two clusters, one closed and one more open, were extracted directly from the material in Figure 3.68. Subsequently, a third cluster was created by merging features of the first two clusters, as illustrated in Figure 3.69.

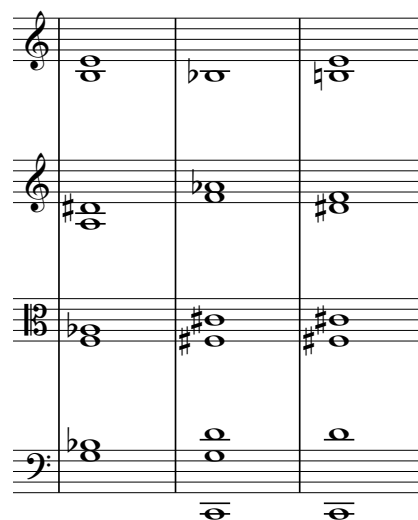


Figure 3.69: Harmonic structures extracted from Ligeti's *2nd String Quartet*

Each vertical harmony from Figure 3.69 can be represented as a scale consisting of two

tetrachords, as shown in Figure 3.70. Tetrachords marked TC2, TC2, and TC3 are shared between the scales in different combinations. The areas marked S1, S2, and S3 correspond to the main three sections of *Union Rose*.



Figure 3.70: *Union Rose*, section tetrachords

Following the scale definitions, the pitch content for the first eight bars was defined, as shown in Figures 3.71 and 3.72, by utilising tetrachords TC1 and TC2. The pitch centre gravitates towards the open string G and related harmonics played in bar 8. The intention was to construct a score that allows for the performance of any bar combinations between different parts. For example, the first violin and cello part in bar 2 should be “playable” with the second violin and viola part in bar 7, with regards to the harmonic texture, rhythm, and pitch. The intention behind this approach was to facilitate the implementation of the dynamic material assignment described above (Figure 1.26), giving both the audience and players the agency to select the instrumentation and music material during the section repeat.



Figure 3.71: *Union Rose*, pitch material, first four bars

Figure 3.73 illustrates an example of how the pitch material from bar 2 for the second violin and viola, shown in Figure 3.71, was implemented in the score. In *Union Rose*, a bar is analogous to a page (i.e. there is one bar per page). Each page consists of 16 beats of proportional notation. The staff layout is consistent for all instruments as defined in *Comprov* (Figure 3.49). The *I. AM.* score (sections A and A') is characterised by the extensive use of bow speed, pressure, and position notation in order to achieve unconventional and unstable sound textures.

The digital audio rendered by MAX patch was first constructed using Logic Pro (1993) software and then mixed in Pro Tools (1989). For the *I. AM.* sections, the audio recording from the *Comprov* session served as a guide track. In the first section, the digital audio mainly

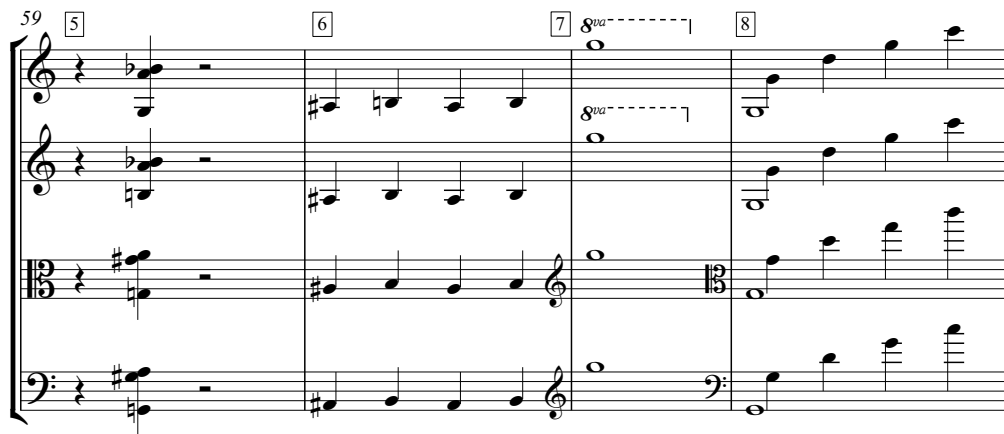


Figure 3.72: *Union Rose*, pitch material, bars 5 - 8

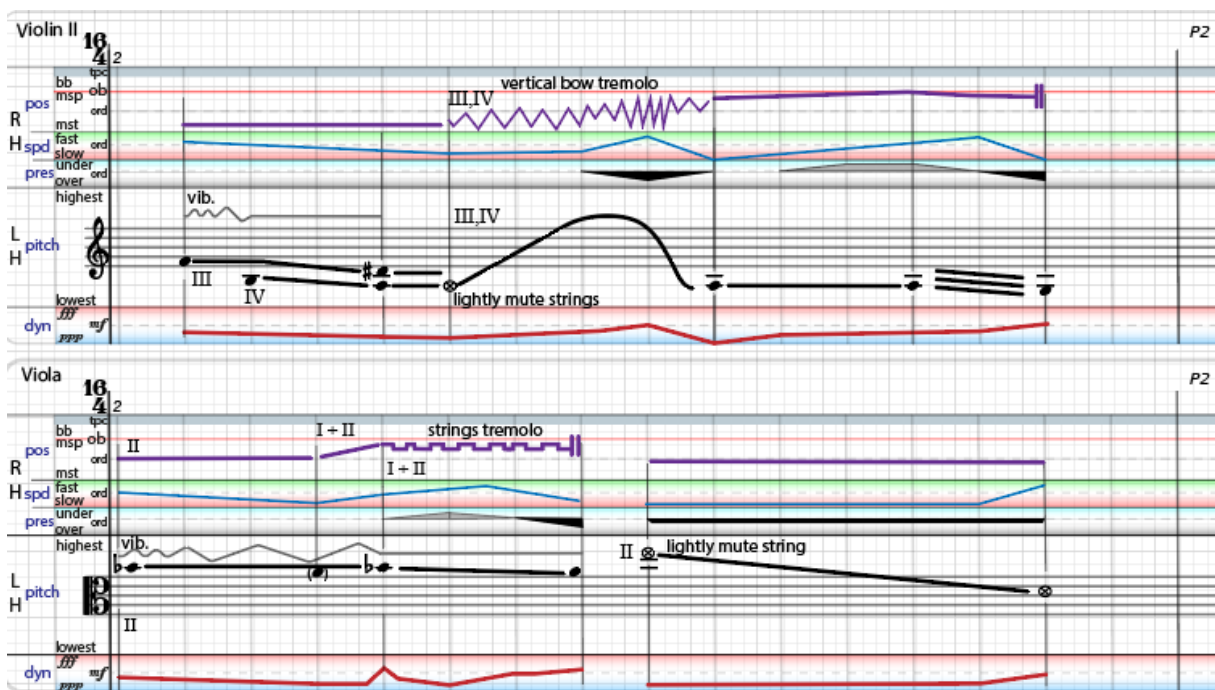


Figure 3.73: *Union Rose*, Page 2 score excerpt

comprises percussive sounds and frequencies supporting acoustic instruments, emphasised either below 100 Hz or above 3000 Hz to compliment the string quartet. One of the technical goals of the project was to achieve precise timing between acoustic and digital sounds. To test this feature, several accented double stops were underlined with percussive digital sounds. The digital audio was cut into WAV files for each bar and assigned appropriate names, such as “UnionRose_b2.wav” for bar 2. This naming convention facilitated easier scripting and scheduling of the digital audio timing.

The first thing the audience sees on their mobile devices after they log in and before the performance starts is the welcome screen (Figure 3.74), showing a black circle stage covering the entire tile grid (Figure 3.62).

Figure 3.74 also illustrates the audience instruction area at the top of the mobile device

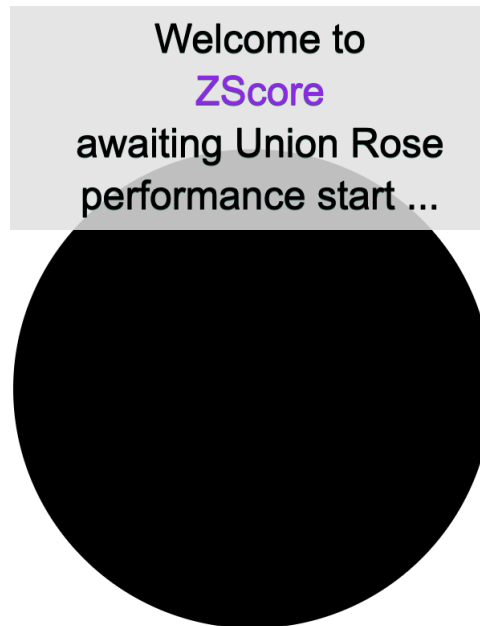


Figure 3.74: *Union Rose*, audience “welcome” screen

screen. This area is frequently utilised in the score to provide information about the piece or issue instructions to the audience regarding any actions that need to be taken. The text can be set directly from the ZScore Control GUI or be populated from a script in the AV part using the following format:

```
web:beat=1:webScore.setInstructions('<line 1>','<line 2>','<line 3>');
```

It is possible to provide up to three lines of text for the instruction area. The font size of the instruction line is automatically adjusted depending on the text length, however, it is advisable to use shorter texts for better readability.

At the start of the performance, the audience view is zoomed in to the innermost tile circle containing words defined in the embedded script for the AV part, bar 1:

```
web:beat=1:webScore.setTileTexts(  
    ['t1-1','t1-2','t1-3','t1-4','t1-5','t1-6','t1-7','t1-8'],  
    ['defective','entitled','helpless','competent','perfect',  
     'failure','confident','worthless']  
)
```

The method `setTileTexts` above maps tiles (e.g. 't1-2') to text (e.g. 'entitled'). The tile identifier (e.g. 't1-2') follows the format `t<row>-<column>`, where `<row>` represents the tile row index and `<column>` represents the tile column index in the two-dimensional tile grid. Figure 3.75 illustrates the audience score representation for section A, bar 1. The green line in Figure 3.75 is the position cursor, which is equivalent to the Web ZScore position tracker. It

indicates the current position within the score and links audience view tiles to music material (i.e. score pages). The colour of the tile frame indicates the composition flow. The currently played tile has a green-coloured frame, whilst the tile to be played next has an orange frame. As the tile is played, it gradually becomes transparent, revealing the shapes behind it.



Figure 3.75: *Union Rose*, audience view, innermost tile circle

Behind the tile frame, only the innermost “centreShape” of the SVG rose defined in Figure 3.61 is visible in the *I. AM.* section. At the beginning of the section play, all constituent SVG paths of the “centreShape” are moved to a random position within the inner circle. A GSAP tween, lasting the full section duration, governs animated movements of all constituent SVG paths back to their original positions, gradually revealing the final look of the “centreShape” at the end of the section.



Figure 3.76: *Union Rose*, audience view, “centreShape” assembly

Figure 3.76 illustrates the assembly of the “centreShape” behind the tile grid during the first part (A) play of the *I. AM.* section. The audience has the ability to click on any part of the tile

grid at any time. The tile selected by the user has a red frame. If the clicked tile row is not currently active, the click will simply rotate the entire row of tiles. However, if the tile is active (is in the currently played row), the audience click count is collected and processed. A click count impacts the background colour of active tiles. The higher the aggregate click count, the darker the tile’s background colour becomes. For example, in Figure 3.76, the tile containing the word “perfect” has the highest click count, whilst the tile with the word “worthless” has the lowest click count. The first part (A) of the *I. AM.* section is played sequentially as written in the score.

When the first part (A) of the *I. AM.* section is completed, the audience view zooms out to reveal the next tile circle (row 2) associated with the section part A’. In the *I. AM.* A’ section, the audience vote decides the order of play, whilst the musicians ultimately decide the instrumentation. In Figure 3.77, the currently played tile has row position 5 and is played immediately after the tile in row position 1. The tile in row 5, containing the word “strong”, is played because it has the highest click count amongst all the tiles in the active row.

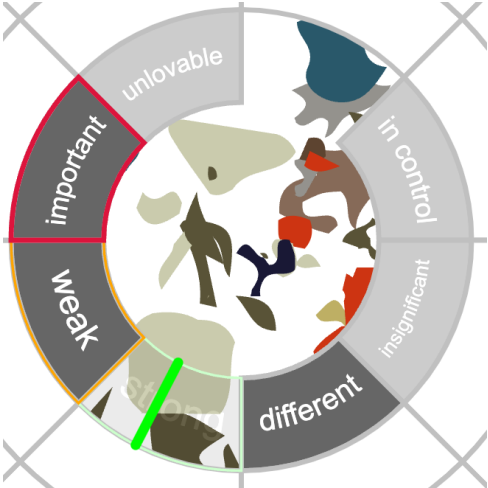


Figure 3.77: *Union Rose*, section *I. AM.* A’ audience view

As mandated by the action model illustrated in Figure 1.30, the context of delayed actions and their corresponding outcomes is outlined through textual instructions and graphical representations of tile sequences over time. The expected timing of action outcomes can be inferred from the green line cursor, which indicates the current position in time.

The page range mapping in the `audienceScoreConfig.yml` file indicates that for tile row 2, pages are assigned sequentially from page number 1 to 8:

```
pageRangeMapping:
...
- tileRow: 2
  tileCols: { start: 1, end: 8 }
  pageRanges: [ { start: 1, end: 8 } ]
  assignmentType: SEQ
```

This means that, in the example shown in Figure 3.77, musicians currently see page 5, although the actual sequential page number is 10 (the first page in the *I. AM. A'* section is 9). In simple terms, for the selected instruments, the content of the page reached by the sequential play is replaced by the content of the page voted by the audience.

The number of instruments participating in *I. AM. A'* section at any point in time is determined by the Randomisation strategy. In bar 7, the AV part contains the following embedded script:

```
sce:beat=12:sce.setRndStrategy([1])
```

This script sets Randomisation strategy to “1”, meaning that only one instrument will participate in the interpretation of the sequential page 9. The Randomisation strategy needs to be set two pages before it takes effect, as the preparatory page in the alternating page layout (Figure 3.10) is displayed one page ahead of the time. The Randomisation strategy initially selects a random instrument from the available pool. This selection is displayed to musicians, as illustrated in Figure 3.57. Musicians can then choose to opt-in or out from playing the next page by clicking the button displaying their part name. The last musician to opt-in before the time window closes wins and is shown the content of the next page selected by the audience. If all musicians opt-out and choose not to participate, then the randomly selected instrument plays the next page, as per the settings defined in the Randomisation strategy.

In bar 8, the AV part contains script:

```
sce:beat=12:sce.setRndStrategy([2])
```

This script sets the Randomisation strategy to “2”, meaning that two instruments will play the sequential page 10. The page content is selected by the audience vote as described above. A chart representing the number of instruments playing each page in the *I. AM. A'* section is displayed in Figure 3.78. The actual instrumentation is ultimately selected by the musicians themselves, apart from page 16, which is played by the entire quartet.

Each page in the *I. AM.* section has a corresponding audio file played by the ZScore Max patch. These audio files, stored on the computer running Max, are loaded into buffer players 1 and 2 and triggered from the score at the start of each page. During the performance, audio files are loaded on the page preceding the play to ensure a smooth audio performance.

For example, the script below loads the file associated with page 3, “UnionRose_b3.wav”, into buffer player 1:

```
max:beat=12:setFile,b1,UnionRose_b3.wav
```

The script is scheduled for execution on page 2, beat 12, to allow for the loading time and ensure an uninterrupted performance. The content of buffer player 1 is then scheduled for play on page 3, beat 1, with the following command:

```
max:beat=1:play,b1
```

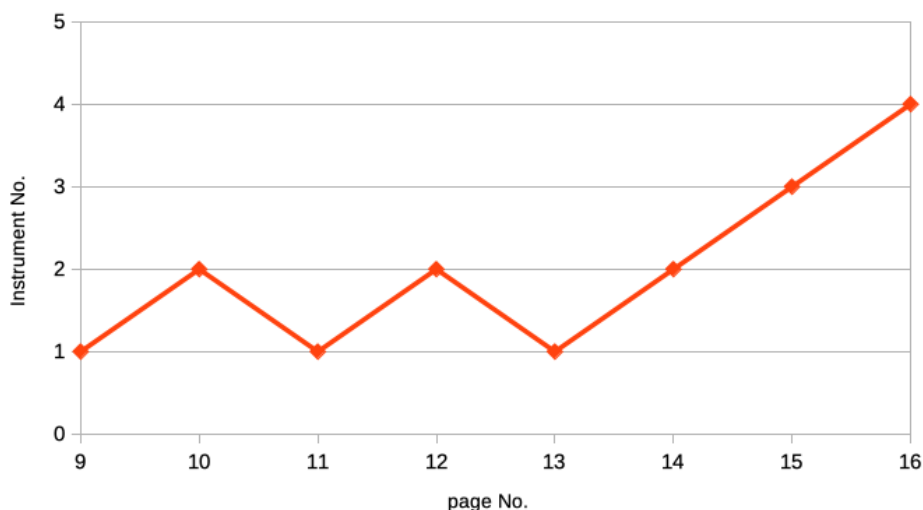


Figure 3.78: *Union Rose, I. AM. A'* section instrumentation

The script above simply executes the “play” command for buffer 1. As “play” is a high priority command in ZScore Max, it is sent directly to the buffer 1 object, bypassing the jsui processing.

Similarly to the alternating pane notation strategy, audio buffers are alternated throughout the play. At least two buffers are required to provide uninterrupted file loading and play in this scenario. To avoid unwanted effects, a played file’s audio amplitude needs to reach zero before the corresponding buffer is reused. Therefore, all audio files used in *Union Rose* have a duration equivalent to the time it takes to play two bars. The audio content is placed only in the first half of the file. The second half of the file can contain some audio aftereffects (e.g. reverb), however, it is usually reduced to silence after several beats of the second bar to allow for buffer reloading without interrupting the audio signal.

In section A’, the file played by the Max patch depends on the page selected by the audience. The embedded script on page 9 of the AV part instructs ZScore to send the selected page to BufGroove 2 with the following command:

```
sce:beat=13:sce.sendMaxMspRndPageUpdates(2) // 2 is the buffer index
```

In addition to BufGroove players, section A’ utilises the ZScore MC Groove Max component. The content of the ZS MC Groove buffer is set in bar 8, just before the start of the A’ section, by the following script:

```
max:beat=11:setFile,groove,UnionRose_b4.wav
```

The identifier “groove” in the script above refers to the ZS MC Groove component. The loaded file “UnionRose_b4.wav” is the same file used in the BufGroove player for bar 4. The usage pattern of ZS MC Groove increases in length gradually during the section, whilst its output is modified in regular intervals by changing the harmonic series of produced audio using the

“subharmonic” and “harmonic” commands. The available harmonic series presets in ZS MC Groove control are as follows:

```

‘off’,
"subharmonic 0.1 1", "subharmonic 0.5 1",
"subharmonic 1.1 1", "subharmonic 0.1 0.5",
"harmonic 0.1 1", "harmonic 0.5 1",
"harmonic 1.1 1", "harmonic 1. 2"

```

An example of the embedded script that modifies the ZS MS Groove harmonic series can be found on page 10 of the AV part:

```
max:beat=7:send,mcGroove.mcgHarmonics,int,2
```

The script above is processed in jsui.js as it is not a high priority command. In this case, the JavaScript function `_send(args)` is invoked, where “args” are: `mcGroove.mcgHarmonics, int, 2`.

The first argument – “`mcGroove.mcgHarmonics`” – is the identifier of the Max live.menu object containing the harmonic series presets defined above. It is a compound identifier where the first part (“`mcGroove`”) is the scripting name given to the ZS MC Groove component in Max, and the second part – “`mcgHarmonics`” – is the scripting name of the Max live.menu placed within the ZS MC Groove component, as shown in Figure 3.79. The second argument – “`int`” – is a Max command setting the index of the selected preset and the third argument – “`2`” – is the preset index to be set. In the preset list above, this index refers to the value “`subharmonic 0.1 1`”, which is sent to the `mc.groove` object.

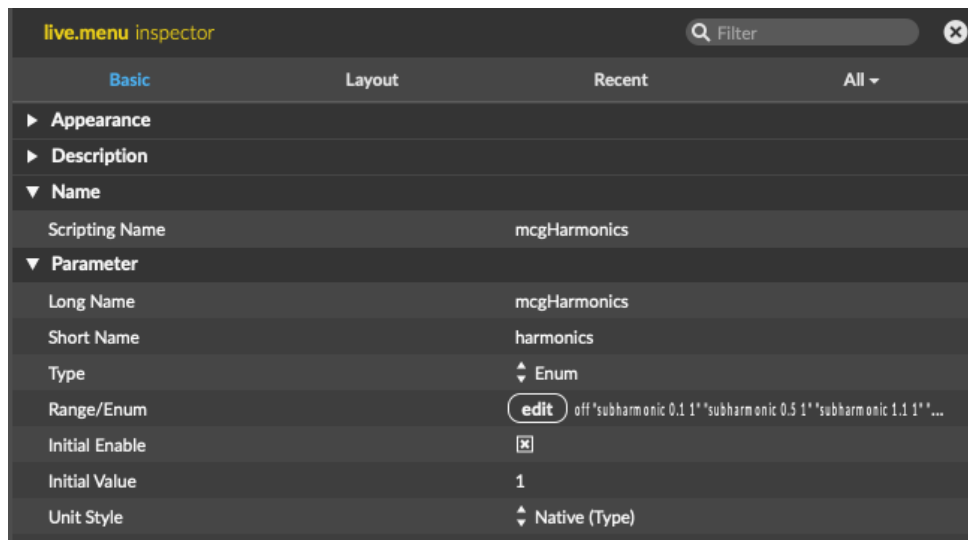


Figure 3.79: Max MC Groove harmonic series preset menu

3.5.2.2 I Believe

This section, representing the second layer of the rose, questions the construction mechanism of human belief systems and explores the nature of human-machine relationships in the age of artificial intelligence. The music material draws inspiration from the avant-garde jazz suite “A Love Supreme” by John Coltrane (1964). In the suite’s first part, named “Acknowledgement”, a four-note theme is repeated multiple times by different instruments and a voice. After many repetitions in the same key, the four-note chant is transposed by a whole tone, as illustrated in Figure 3.79. Coltrane’s handwritten notes on the original score (1964) indicate the religious subtext of the piece, sketching out the form of part one as a musical prayer.



Figure 3.80: Coltrane’s “A Love Supreme” four-note chant

In *Union Rose*, the four-note pattern of ascending minor third and perfect fourth intervals from “A Love Supreme” is used to construct the pitch material of the *I Believe* section. Figure 3.81 outlines the pitches used in the first two bars of the section.



Figure 3.81: *Union Rose*, *I Believe* section pitch material

In contrast to the textures used in the *I. AM.* section, *I Believe* is characterised by clearly intonated sustained notes. The first two bars serve as a contrasting introduction to the section, where the material in the first bar (Figure 3.82) is played as loudly as possible, whilst the second bar requires a quiet flautando (Figure 3.83). Accents are provided by bow speed modifications, similar to the techniques used in the *I. AM.* section, thus maintaining continuity in the produced sound character.

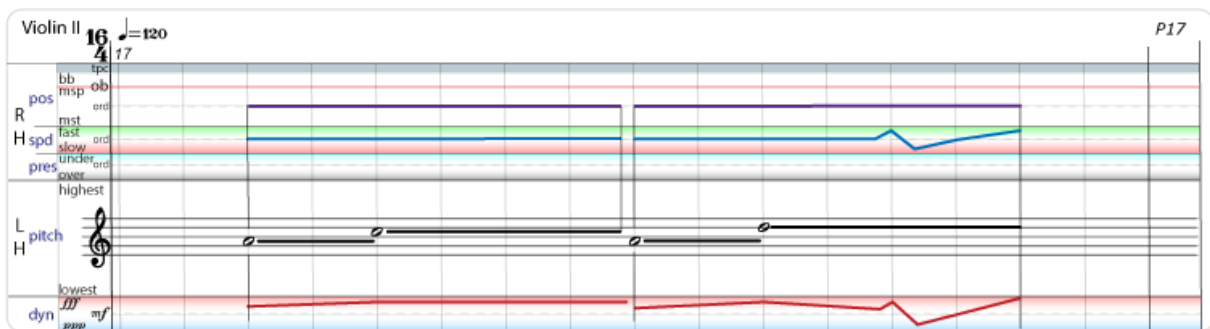


Figure 3.82: *Union Rose*, Violin 2, section B first bar

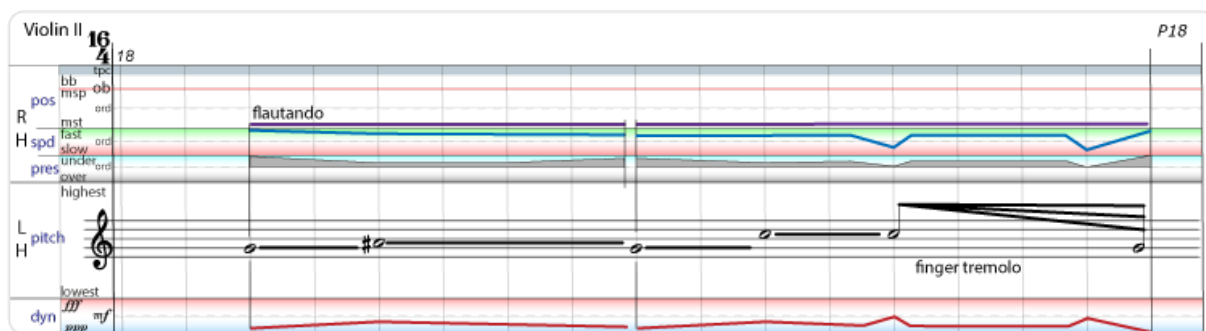


Figure 3.83: *Union Rose*, Violin 2, section B second bar

The rest of the B section notation is derived from a development of the initial four-note pattern, with the music material’s density gradually increasing. This culminates with a quaver-length theme that is shifted in time and pitch between different instruments on the last two pages of the section.

In section B, the audience score representation is zoomed out to the next tile ring (row 4) whilst the SVG rose “innerCircle” group is made visible in addition to the “centreShape” group. Similarly to section A (*I. AM.*), all SVG parts belonging to the “innerCircle” group are initially moved to random locations within the outer stage. The SVG paths are then gradually moved to their original positions during the *I Believe* section play. The section B tiles and corresponding pages are played in sequential order, as written [P17, P24].

In addition to the acoustic texture contrast, the digital audio in section B is produced in a very different way compared to the previous section. Each bar starts with a drum kick reminiscent of a digital heartbeat. The remainder of the digital audio texture is filled with synthetic voices uttering the phrase “I believe in”, accompanied by a word assigned to the currently played page. The page text mapping is set on page 17 of the AV part.

The synthetic voices used in section B were generated using copyright-free text-to-speech websites such as text2speech.org (2018) and fromtexttospeech.com (2020). To create the synthetic voices, a text file containing all the words associated with the active tiles and a mixture of sentence structures starting with “I believe” were submitted to the text-to-speech engines. Different configurations of available voices and speech speeds were experimented with to achieve the desired vocal effect for each word.

The resulting mp3 files containing synthesised speech were imported into Pro Tools and processed to conform to the structure of section B. To emphasise the recent emergence of artificial intelligence and to provoke questions about the process of belief system construction, voices generated by older, robotic-sounding synthesis engines were preferred over those that sound more natural. Furthermore, a combination of different accents and genders was used in the voice synthesis to underscore the universal nature of the themes explored in the piece. The voices were strategically positioned at varying depths and locations within the stereo field, acting as an extension of the audience space boundaries.

In section B', the audience score representation is zoomed out to reveal the next ring of tiles (row 5), shown in Figure 3.84. The tile text mapping for row 5 is set in the AV part, page 25, as follows:

```
web:beat=1:webScore.setTileTexts(
    ['t5-1', 't5-2', 't5-3', 't5-4', 't5-5', 't5-6', 't5-7', 't5-8'],
    ['conspiracy', 'equality', 'monarchy', 'global warming',
     'free market', 'sovereignty', 'europeanism', 'democracy']);
```



Figure 3.84: *Union Rose, I Believe*, section B' audience view

The intention is to modify available words based on current global trends at the time of a performance. These words, like any other score script, can be edited in either the Adobe Illustrator score or a dedicated BeatInfo csv file. For *Union Rose*, this file is named *Union_Rose_BeatInfo.csv* and it resides within the score's "rsrc" folder. Changes made in Adobe Illustrator require exporting the score through the ZScore tools plugin before a performance, whilst the BeatInfo file can be manually modified at any time up to the moment the score is loaded into the ZScore system.

Similar to section A', the order of play in section B' is governed by the Randomisation strategy as set in the *strategyConfig.yml* file for *Union Rose*:

```
- range: { start: 25, end: 32 }
  isRndActive: true
  selectionRange: [ { start: 17, end: 24 } ]
  instruments: [ all ]
```

The flag “isRndActive” indicates that the Randomisation strategy is active for the section B’ page range [25, 32]. The music material for section B’ comes from the section B page range [17, 24]. By default, active tiles in the audience score representation are sorted by the number of clicks. When the scheduling engine requests the next page to play, the web score engine returns the page corresponding to the tile with the highest number of votes. However, this behaviour can be overridden by setting the flag “isSortByClickCount” to false for active rows in the method:

```
setActiveRows(int[] rows, boolean isSortByClickCount){}
```

The embedded script on page 25 in the AV part sets this flag, forcing the row 5 tiles to play in a sequential order:

```
web:beat=1:webScore.setActiveRows([5], false)
```

The instrumentation selection in section B’ follows the same process as explained above for section A’. The number of instruments used per page follows the same curve as shown in Figure 3.78. As in section A’, musicians have the final say in the instrumentation choice.

For the first time in the composition, in section B’, audience members can join the performers by producing sounds either on their mobile devices or by saying words of their choice. Similar to the sound textures created by the Max patch in section B, the audience can generate speech patterns by clicking on active tiles in the audience score visualisation. When a click event occurs on any active tile in section B’, it produces a speech utterance consisting of a sentence starting with “I believe in” followed by the word associated with the selected tile. The mobile device speech is rendered through the Web Speech API, which is available in almost all modern browsers installed on mobile devices. The choice of available voices depends on the device’s operating system and, in some cases, the manufacturer. The audience score implementation selects a random voice from the list of available voices every time a tile is clicked, thereby avoiding any bias.

From bar 28 onwards, in addition to the speech produced by the mobile devices, audiences are encouraged to say words mirroring their tile choice. Initially, they are instructed to say the words quietly, and as the section progresses, they are encouraged to gradually increase their volume towards the end of the section. Clear instructions regarding the actions and timings are displayed on the top of the mobile device screen to guide the audience on when and how to participate.

The merging of mobile device utterances and natural speech was intended to create an electro-acoustic choir, where individual voices blend together into the overall speech texture, combined with the amplified digital audio. This effect is reminiscent of a heterogeneous group prayer, where diverse voices come together in a unified expression.

The introduction of granulators in section B’ further enriches the sound texture, both on mobile devices and in the Max audio output. The file “UnionRose_m1”, was specifically produced for the granulator textures in this section and contains generic utterances rendered by

the same synthetic voices used in section B. To ensure optimal performance and faster downloads on mobile devices, the granulators use smaller mp3 versions of the same file, whilst the Max patch utilises the high-quality WAV file format for more precise control and sound manipulation.

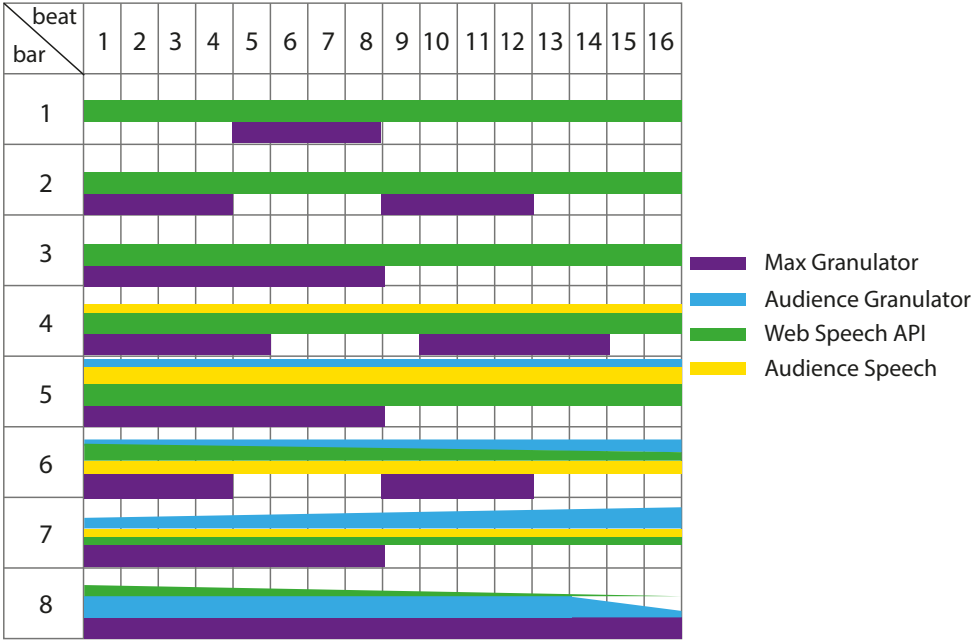


Figure 3.85: *Union Rose, I Believe*, section B' speech texture components

Figure 3.85 displays a table illustrating the speech texture components in section B'. The texture is formed by four main components: the Max Granulator played through a PA system, the Audience Granulator and Web Speech running on the audience's mobile devices, and finally, the Audience Speech spoken by members of the audience. The rows in the table represent section B' bars, whilst the columns represent beats within each bar. To understand the texture's development over time, the table should be read from left-to-right and top-to-bottom. The thickness of the line in each cell represents the relative volume of the corresponding audio component. In a live performance, the ratio of volume levels for each component depends on the size of the audience and their level of participation in the speech texture.

Web ZScore offers a variety of utility objects that enable the gradual modification of any JavaScript object parameter over time. These utility objects include RampLinear and RampSin, which provide linear and sinusoidal parameter interpolation, respectively. Additionally, GsapRampLinear utilises GSAP tween and function callbacks for parameter value calculation. ParameterOscillator is another useful utility that can internally deploy different Oscillator types to perform parameter evaluation on demand. The available Oscillators include:

```
'SAWTOOTH', 'SINE', 'SQUARE', 'TRIANGLE', 'UP', 'DOWN', 'RANDOM'
```

For example, the Audience Granulator volume changes in Figure 3.85 are executed from the embedded script in bar 30 (section B' page 6):

```
web:beat=1:webScore.granulatorRampLinear('masterGainVal', 0.5, 8000);
```

The script above deploys the RampLinear JavaScript object in audience web clients to gradually change the granulator parameter “masterGainVal” from its current value to “0.5” over a duration of 8 seconds (8000 milliseconds) using linear interpolation.

3.5.2.3 I Want

The *I Want* section, representing the outer circle of the rose (Figure 3.61), examines the relationship between an individual and the relentless consumerism prevalent in the late 20th and early 21st century. The music material seed comes from the popular song “Money, Money, Money” by the Swedish group ABBA (written by Benny Andersson and Björn Ulvaeus), which was released on 1 November 1976. The song explores a state of mind engrossed in self-pity, professing that even “a little money” would make a difference in an unjust world, eventually reaffirming the belief that money is the key to perpetual happiness (“Always sunny in the rich man’s world”). The “Money, Money, Money” chorus melody is shown in Figure 3.86.



Figure 3.86: “Money, Money, Money” chorus melody

The three-note ascending pattern from the first bar of the “Money, Money, Money” chorus has been extensively utilised in section C, albeit by transforming both time and pitch intervals. In contrast to the long sustained pitches in section B, the entire C section consists of short percussive or plucked pizzicato notes. An example of the section C notation is shown in Figure 3.87.

Figure 3.87: *Union Rose*, Viola part, section C notation

A notation separation allowed for different timings of the left and right hand actions, resulting in unstable pitch movements. The instability of the pitch was underlined by the frequent repetition of the repeated F# glissandi “soundbite” shown in bar 38, beat 1 (Figure 3.87). The rhythmical pattern timings and pitch movements gradually converge to play the melody in unison on the last page of the section C, bar 40, beat 4 shown in Figure 3.88.

In the *I Want* (C) section, the rose’s “outerCircle” group (Figure 3.61) is finally activated, making all rose shapes visible behind the cover tiles. The behaviour of SVG elements and the

Figure 3.88: *Union Rose*, Cello part, section C, last page

order of play is consistent with the rules established in sections A and B. However, Section C' of *I Want* differs in several ways from previous sections. The rate of activity, both visual and audible, reaches its peak. The audience score representation is zoomed out to reveal the remaining two active rows of tiles (7 and 8), as shown in Figure 3.89.

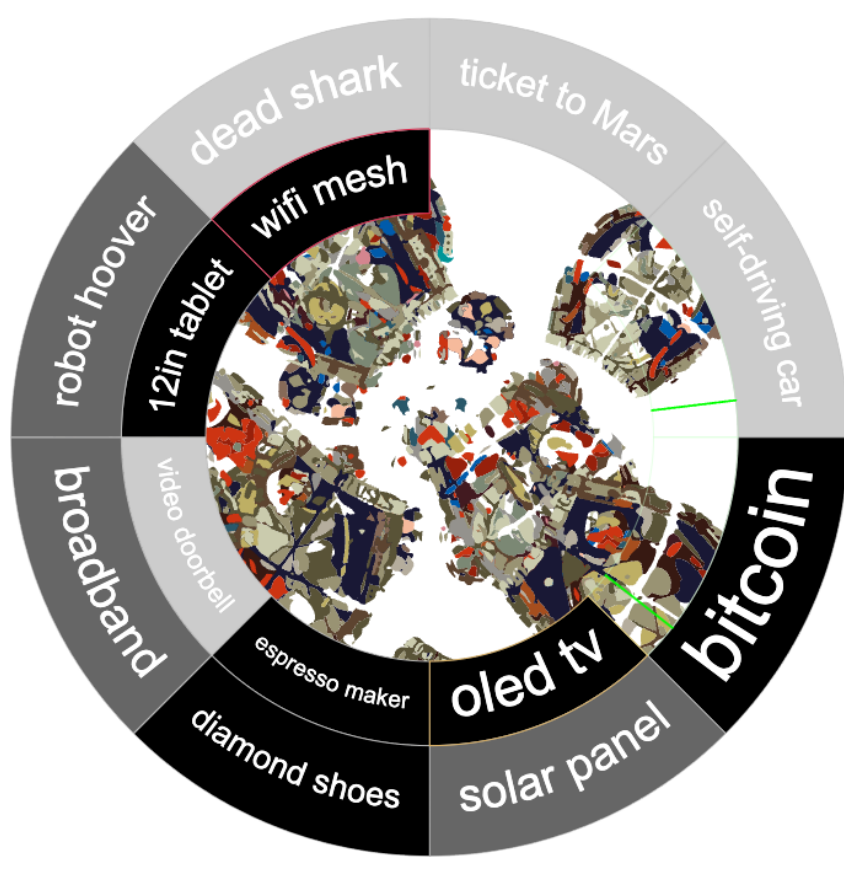


Figure 3.89: *Union Rose*, *I Want*, section C', audience view

The page range used in the section C' Randomisation strategy is expanded to include all unique score pages in the score from sections A, B, and C. The file `audienceScoreConfig.yml` contains the same page range configuration for tile rows 7 and 8 as follows:

```
- tileRow: 7
  tileCols: { start: 1, end: 8 }
```

```

pageRanges: [
  { start: 1, end: 8 }, { start: 17, end: 24 }, { start: 33, end: 40 }]
assignmentType: RND

```

The Randomisation strategy configuration for the page and instrumentation selection is configured to utilise multiple unique pages played by one or more instruments. Figure 3.90 illustrates instrument and page Randomisation strategy selection for each bar of section C'. The audience score representation reflects multiple page play by activating green play cursors on all tiles associated with the currently played pages. Similarly to single tile play, all currently played tiles dissolve gradually during the page play.

	unique pages			
bar	1	2	3	4
1	■			
2	■	■		
3	■			
4	■	■		
5	■			
6	■	■	■	
7	■	■	■	
8	■	■	■	■

Figure 3.90: *Union Rose, I Want*, section C' instrument-page randomisation table

The coloured squares in Figure 3.90 represent the number of instruments that play a particular page. For example, the table row for section bar 6 (equivalent to the elapsed bar 46) contains two green and one orange square. This indicates that two instruments (green squares) play the same random page (1) and one instrument (orange square) plays a different random page (2). This configuration is set by the embedded script on page 44:

```
sce.setRndStrategy([2,1])
```

The randomised selection follows the same logic as shown in Figure 3.54. Musicians ultimately decide who plays each bar by utilising the selection features depicted in Figure 3.57. In the last bar of section C' (section bar 8), all four members of the quartet play different random pages selected from all available *Union Rose* sections.

The ZScore Max BufGroove components are configured to play audio files associated with the first randomly selected page. The page-file selection is communicated to Max by the embedded script:

```
sce.sendMaxMspRndPageUpdates(<buffer no>)
```

In addition to BufGroove audio, the ZScore Max components ZS MC Groove and Granulator are also active in section C'. The audio file associated with the very beginning of *Union Rose* ("UnionRose_b1.wav") is loaded into ZS MC Groove, whilst the Granulator plays the

same file associated with section B' ("UnionRose_m1.wav"), hinting at the upcoming material recapitulation.

The audience score representation produces speech patterns on any active tile click, similarly to the behaviour in section B'. However, this time the speech utterance starts with the words "I want". The tile speech is set on page 41 of the AV part with the embedded script:

```
web:beat=1:webScore.setSpeechText('I want @TILE_TEXT@.')
```

The token "@TILE_TEXT@" in the command above is replaced with the content of the tile text.

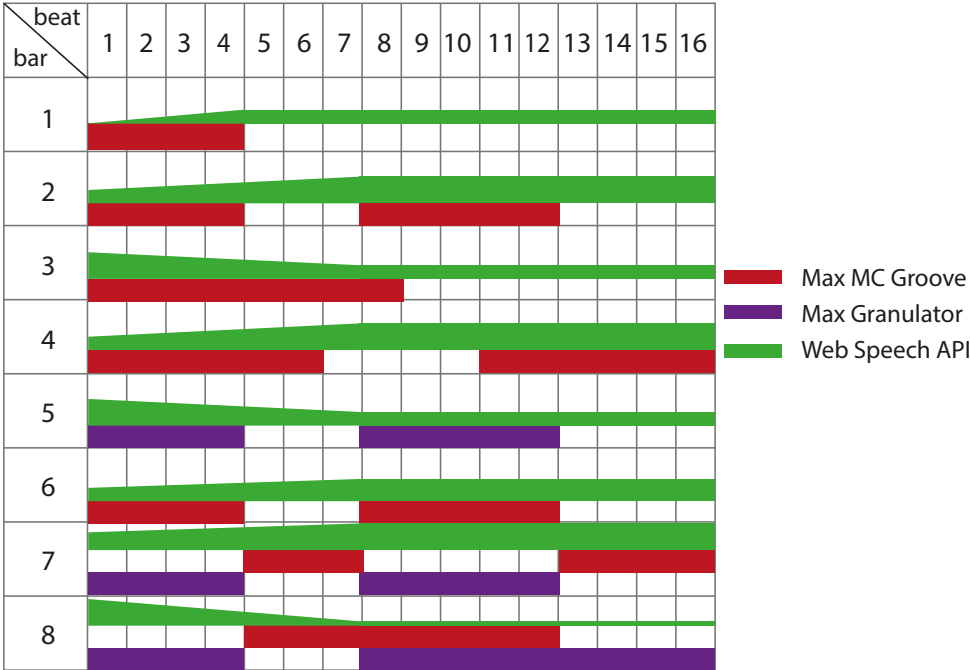


Figure 3.91: Union Rose, I Want, section C' digital audio texture

The overall digital sound texture in section C' is illustrated in Figure 3.91. It should be read in the same way as the table presented in Figure 3.85. The relative thickness of the line represents the individual component volume. The overall sound texture depends on the number of audience members and their actions during the section play.

3.5.2.4 Coda

Coda starts with the recapitulation of the last four pages [P4, P8] of section A. After the initial literal recapitulation, the second half of the Coda [P53, P56] incorporates a randomised recapitulation, where the material is randomly selected from the entirety of section A. The recapitulation Randomisation strategy is configured in strategyConfig.yml as follows:

```
- range: { start: 53, end: 56 }
  isRndActive: true
```



```

selectionRange: [ { start: 1, end: 8 } ]
instruments: [ all ]

```

The instrumentation for the last four pages is also determined by the randomisation algorithm, with the specified number of instruments per page as follows:

```
[Page Number - Number of Instruments]: [53 - 2, 54 - 1, 55 - 2, 56 - 1]
```

Therefore, the piece always concludes with a single randomly selected instrument playing a randomly selected page from section A.

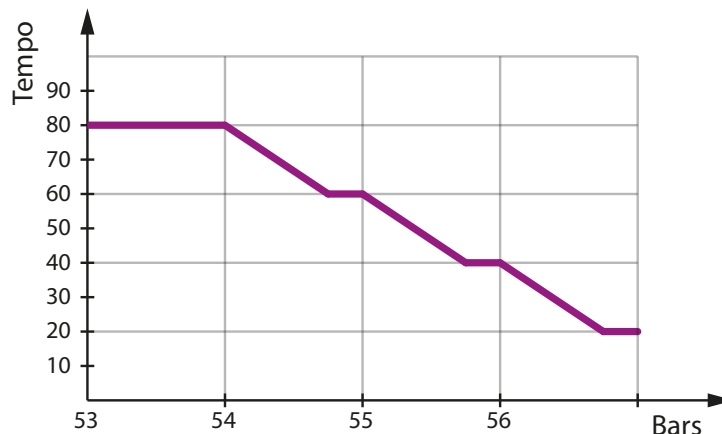


Figure 3.92: *Union Rose*, *Coda* tempo curve

The last four pages of the *Coda* bring the performance to a close by gradually diminishing the density and tempo of the piece. The tempo curve applied in the last four bars of the *Coda* is displayed in Figure 3.92. The gradual ritardando is initiated by the scripts in the AV part. For example, on page 54, the Scripting Engine script executes a timed action on the server:

```
sce:beat=1:sce.timedAction('tempo', 60, 12);
```

The “timedAction” function takes three arguments: the parameter name (‘tempo’), the end value (60 bpm), and the duration expressed in a number of beats (12). When the script is executed, the server-side Scripting Engine sends the tempo change event on every beat, adjusting the current tempo by the appropriate value determined through linear interpolation. As visualised in Figure 3.92, the tempo in bar 54 changes from 80 bpm to 60 bpm in 12 beats, taking $\frac{3}{4}$ of the bar duration.

The ZScore MAX Granulator, loaded with the *Coda* specific audio file (*UnionRose_c3gr.wav*), is triggered to play on the first beat of the last four bars. This audio file contains the same synthetic voices used in the section *I Believe*; however, this time, the voices only repeat the phrase “I am a”. The Granulator controls are set for a gradually slower sweep, resulting in the occasional comprehensible rendering of the words. Unlike previous Granulator participations that use continuous play, the *Coda* scripts use one-off Granulator triggers with the help of the Max line~ object and the gain envelope defined in the trigger scripts. For example, the last Granulator trigger is called in bar 55 of the AV part:

```
max:beat=13:setLine,granulator,0.@C@ 1. 1000 1. 15000 0. 16300
```

The token @C@ is replaced by a comma on the server before being sent to Max. A comma cannot be used in the score directly as the score file uses a comma delimited format. The command above plays the Granulator with the gain envelope: 0 to 1 in 1 second (1000ms), sustains at gain 1 for 15 seconds (15000ms), and gradually reduces gain to 0 in 16300ms to coincide with the end of the piece.



Figure 3.93: *Union Rose*, *Coda* audience view, selected tiles animation

The audience score representation in *Coda* serves as a visual recapitulation of the user’s behaviour during the piece. To start with, audience members finally see the entire rose on their mobile devices. The recapitulation starts with an animation bringing back all tile shapes that were previously selected by the user in each section, as illustrated in Figure 3.93. The last selected tile shapes, coloured in red, gradually move back into their original positions from beyond the visible boundaries. This visual recapitulation aligns with the musical content and allows the audience to reflect on their interactions with the piece.

In bar 53, all rose shapes: “outerCircle”, “innerCircle”, and “centreShape” “explode” one by one into their constituent SVG paths. Only “centreShape” fragments, corresponding to the *I. AM.* section, end up randomly scattered across the stage visible on the mobile device.

The circular stage, which has been transparent since the beginning of the piece, is gradually returning back to black, resembling the starting point illustrated in Figure 3.74. However, this time the stage has the tile selections unique to each audience member layered on top, representing their journey through the piece (Figure 3.94). All choices are interlinked with thin

red lines, creating a visual representation of the individual paths taken by each audience member throughout the performance.



Figure 3.94: *Union Rose, Coda*, audience view end screen

In the penultimate bar 55, the following question is displayed on the audience's mobile devices:

‘‘The choices you’ve made?’’

The question raises doubts about the sources of human agency and encourages contemplation about free will, as well as the potential impact that social interactions and technology may have on its formation.

3.6 Socket Dialogues

For:

Multiple Instruments

Max

and Audience

Duration:	10 - 30 minutes
Links:	Full Score ZScore Package Workshop 1 video recording Workshop 2 video recording

This composition explores the democratisation of music-making roles through the use of network technology (3.6.2). The piece consists of a number of musical dialogues where musicians, audience members, and the conductor can choose to perform different roles. Each dialogue role is associated with specific music material. Musicians select their dialogues and the order of play at the beginning of a performance (Figures 3.98 and 3.99), whilst they can change the role they wish to perform during each dialogue (Figure 3.101).

The score is written for any number of musicians and any instrumentation. To enable this notational flexibility, a new type of dynamic generic notation was developed (3.6.2.2). Audience members can provide instant visual and auditory feedback to all other participants throughout the dialogues via their mobile devices (Figure 3.109), leading to continual adjustments to the performance output (3.6.2.5). Beyond shaping the performance, the audience can also take on the role of a performer by playing audio through their mobile device instrument (3.6.2.6) in designated audience-led dialogues (*Interlude*, 3.6.3.6). The composer/conductor can also become a performer during *Interludes* by engaging with ZScore Max patches (Figure 3.64). Whilst the Max patches developed for *Union Rose* were reused, the audio file content was created specifically for each dialogue. All these features combine to create an environment for meaningful performative involvement by all participants.

3.6.1 Socratic Dialogues

Socket Dialogues' name is a pun on Plato's Socratic Dialogues (Plato, Jowett edition 1892), a collection of around 35 preserved literary works written in the 4th century BCE that utilise the Socratic method to uncover and scrutinise underlying beliefs and assumptions about various philosophical subjects. The word Socket refers to an endpoint in a communication between computer programs over a network (Winett 1971).

The Socratic method involves a series of questions and answers designed to stimulate critical thinking and draw out ideas about the discussion topics. Plato's dialogues start with the assertion of a thesis by an interlocutor – a person arguing the point on a given subject. Socrates, always the main protagonist of Plato's dialogues, then examines the thesis, questioning its validity and challenging the interlocutor's beliefs. This initiates an argument between the dialogue participants. After an extensive discussion, the dialogue outcomes provide new insights into

the debated subject, the validation or falsification of the thesis, or end up in aporia, a state of puzzlement and confusion.

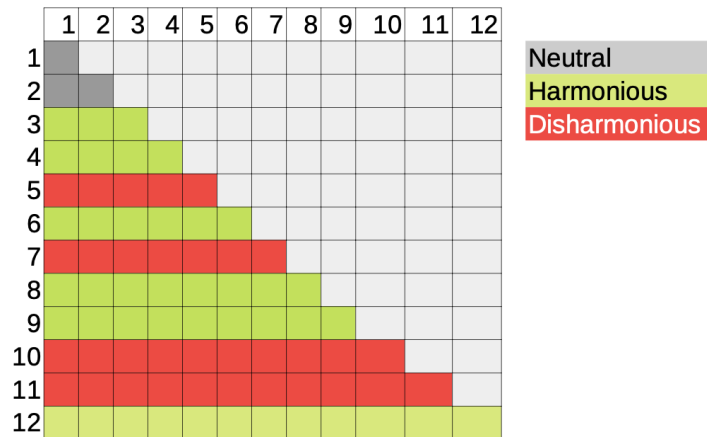


Figure 3.95: Plato's dialogues scale

Kennedy (2010) argues that the number twelve “has some architectural importance” in Plato’s dialogues, based on the analysis of speech lengths, position of speeches within dialogues, the location of significant turns and absolute lengths of the dialogues. He concludes that Plato uses this discovered twelve-part stichometric structure as a form of a musical scale. The scale described in terms of integer ratios consists of intervals where “the third (1:4), fourth (1:3), sixth (1:2), eighth (2:3), and ninth (3:4) notes on the twelve-note scale will best harmonise with the twelfth” (Kennedy 2010, p. 17). It can be deduced that the other notes from the twelve note scale are dissonant or “disharmonious” when played together with the twelfth note. This principle of the scale stability can be visualised as shown in Figure 3.95.

In *Socket Dialogues*, the method and twelve-part structure of Plato’s works serves as the seed concept for the further compositional development. The starting point was the definition of twelve elements of music that first came to mind (Figure 3.96). These elements were then mapped to the circle of fifths counter-clockwise, thereby becoming the circle of fourths. The circle of fifths is a way of organising the twelve chromatic pitches that also serves as the bedrock of Western classical music functional harmony.

	1	2	3	4	5	6	7	8	9	10	11	12
Dialogue	pitch	rhythm	tempo	improv	melody	harmony	dynamics	timbre	space	intervals	texture	interaction
Centre pitch	G	C	F	Bb	Eb	Ab	Db	Gb	B	E	A	D

Figure 3.96: Aspects of music mapped to the circle of fourths

From this list, five aspects of music were chosen as the most interesting for a potential musical dialogue: pitch, rhythm, melody, improv, and timbre. A musical dialogue was created for each of these aspects based on the twelve-part division of Plato’s works. Additionally, interlude dialogues can be performed between the aspect dialogues by all participants as guided free improvisation.

3.6.2 Objective: Music-making Role Democratisation

A musical score containing symbolic notation for more than one instrument is usually divided into instrument-specific parts by the composer. Each score part contains symbolic notation with the appropriate clef, transposition, and range for the respective instrument. Musicians are generally expected to accept their parts as immutable.

Socket Dialogues is an attempt to change this traditional paradigm and provide musicians with the freedom to choose their parts, materials, and the order of performance. Importantly, musicians are also given the agency to decide not to play certain parts if they so wish. The audience can provide direct feedback at any point during the performance through visual and audible cues. Furthermore, audience members can initiate and shape the performance by producing audio from their mobile devices at certain sections. ZScore tools allow a composer or a conductor to become an active performer alongside the audience and musicians.

3.6.2.1 Dialogue Roles

Socket Dialogues' score is not written for specific instrument parts but rather for specific participation roles that can theoretically be played by any instrumentation and by any number of musicians. An interlocutor, in this case a musician, can choose to: Present, Concur, Dissent, or Abstain in each dialogue. The Presenter introduces the dialogue's musical thesis, whilst other participants agree (Concur) or disagree (Dissent) with the musical argument. Musicians can also decide to give up (Abstain) during a dialogue if they are not a Presenter. The score is distributed to musicians based on the selected participation role. The only restriction regarding the instrumentation is that there has to be at least one Presenter for each dialogue.

The first screen musicians see when the *Socket Dialogues* score is loaded is the instrument transposition selection (Figure 3.97). As all roles can be played by any instrument, the score needs to be presented in the correct transposition. The transposition method and generic score presentation is explained in more detail below.

Once musicians select the appropriate transposition for their instrument, they are given a choice of the dialogues they can choose to present (Figure 3.98).

Each connected musician can then choose to present any of the dialogues on a first come, first served basis. Once the dialogue is assigned to a presenter, it disappears from the list of available dialogues, as shown in Figure 3.99.

In Figure 3.99, the musician has selected to present timbre, melody, and pitch dialogues. The selection continues until all active dialogs have an assigned presenter.

At the same time, the performance Control GUI gets populated with Presenter selections, forming the dialogues' order of play. The list of dialogues on the right in Figure 3.100 shows the order of dialogue selections by the Presenters, with the current dialogue being highlighted (pitch) and the next being set to melody. The conductor has the ability to change the order of play at any point by selecting the next dialogue to play in the Control GUI list. Each web score

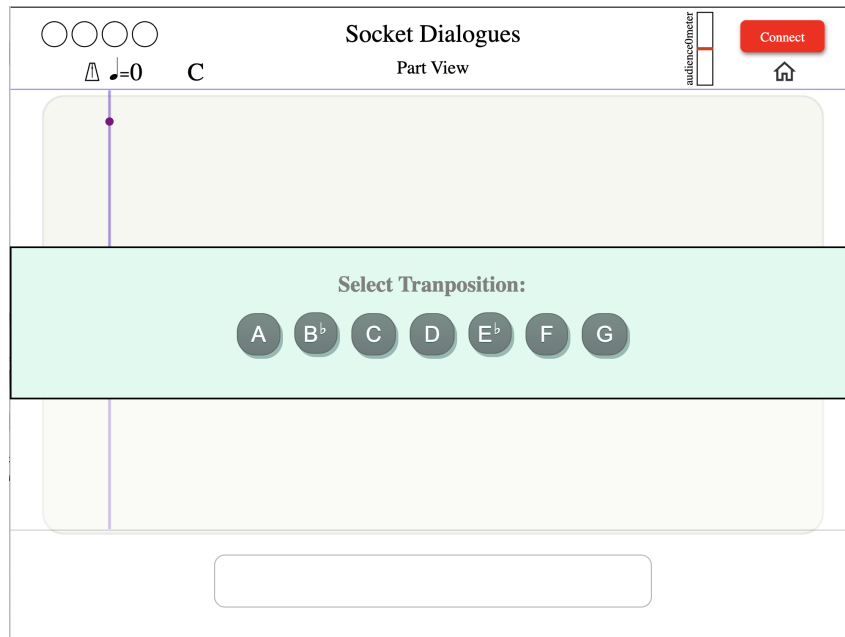


Figure 3.97: *Socket Dialogues*, web score, instrument transposition selection

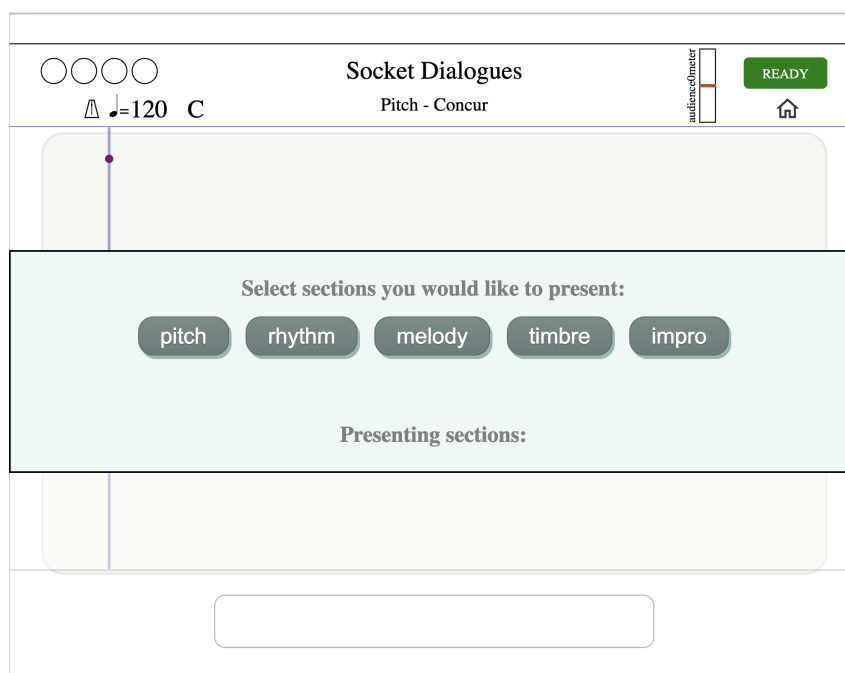


Figure 3.98: *Socket Dialogues*, Presenter selection

client is given an id visible in the Owner column on the left side in Figure 3.100. This id can be used to identify Presenters at any point if required.

Once the dialogue play begins, all connected musicians, apart from the Presenter, can see a choice of actions representing possible responses. The bottom pane buttons in Figure 3.101 illustrate the available actions: Concur, Dissent, or Abstain. Clicking on any of the available buttons visualises the score for the selected role. The currently selected role is highlighted in green. For the case illustrated in Figure 3.101, it is Concur, which also serves as the default

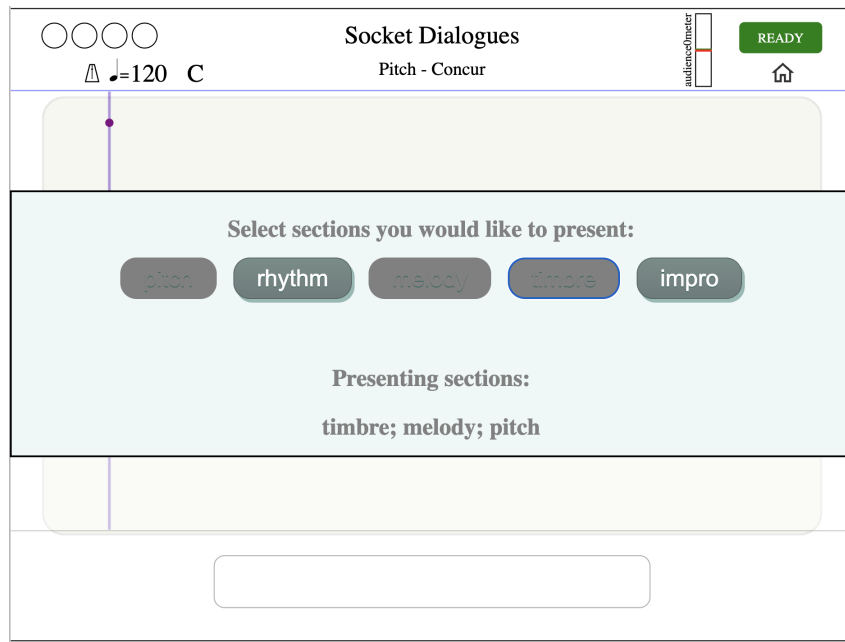


Figure 3.99: *Socket Dialogues*, Presenter selection in progress

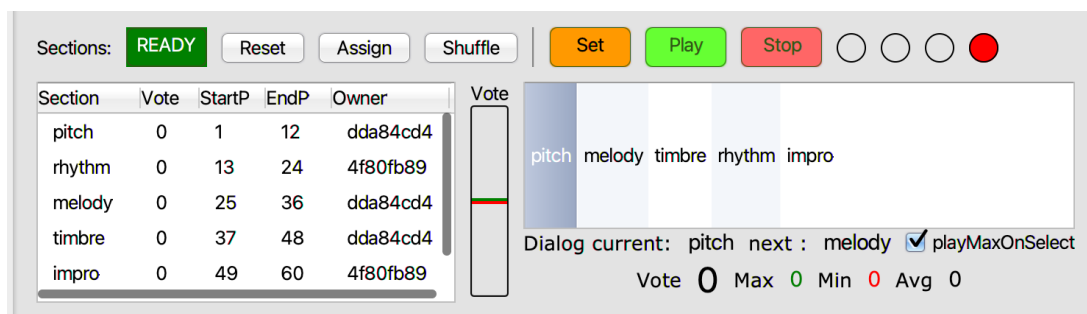


Figure 3.100: *Socket Dialogues*, Control GUI order of play

response. Apart from the Presenter, musicians can change their role at the beginning of every page, giving them the ability to play different material during the dialogue. The overall sound texture's balance may vary depending on the number of musicians involved.

In order to verify all possible vertical structure combinations, all dialogues were first provisionally written out in Sibelius notation software. Possible simultaneous part combinations include: solo Present, duo Present/Concur or Present/Dissent, and trio Present/Concur/Dissent. Each part can be performed by any number of musicians.

3.6.2.2 Generic Instrument Notation

In order to provide notation playable by any instrument, it was necessary to invent a transposition strategy suitable for the mixed notation style. The staff layout used in *Socket Dialogues* is a simplified version of the layout used in *Union Rose* (Figure 3.102).

The information in the top left corner of the staff provides details about the displayed notation, including the name of the currently played dialogue and the musician's role in the

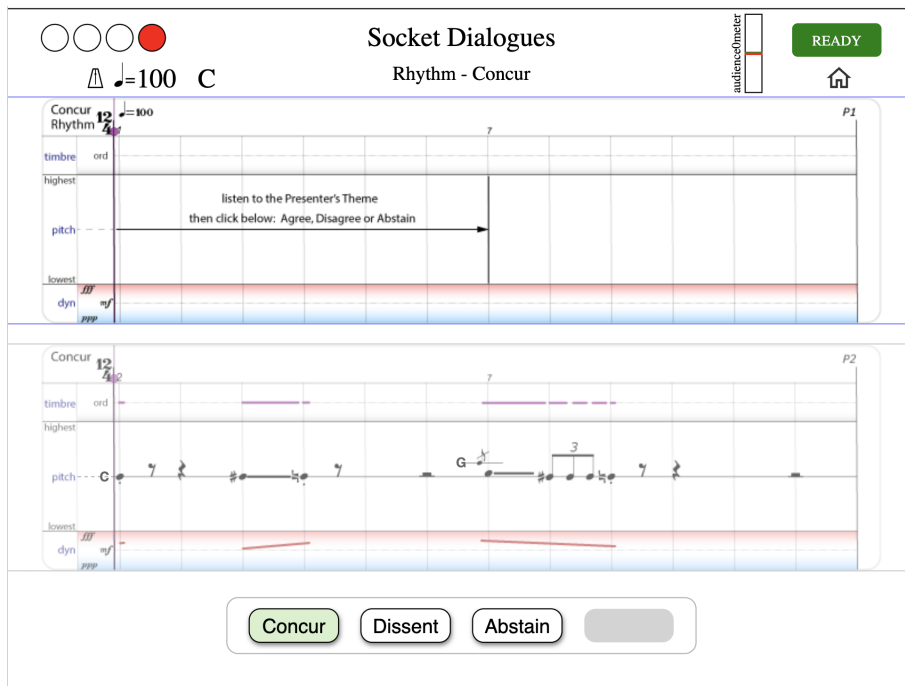


Figure 3.101: *Socket Dialogues*, web score role choice

Stave notation

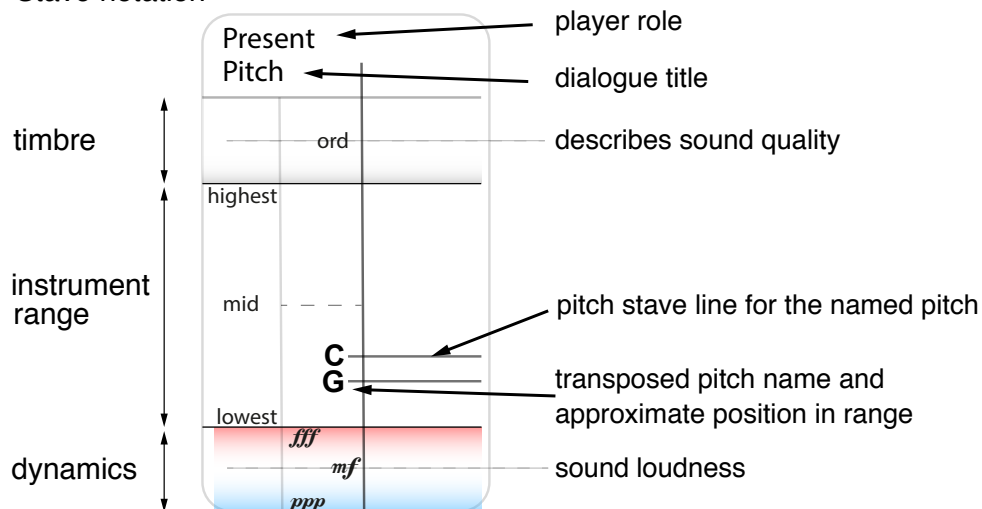


Figure 3.102: *Socket Dialogues*, stave layout

dialogue. The rest of the stave is divided vertically into three main areas: timbre, pitch, and dynamics information. The dynamics area works in the same way as in *Union Rose*. The timbre area of the stave is used to describe the required quality of sound in generic terms by using words and colours, as illustrated in Figure 3.103. The sound quality terms are borrowed from the strings and wind instruments' vocabulary, however, they could be translated into any instrument's sound palette as an indication of the sound variation within available boundaries.

The innovative pitch notation is based on named pitch lines. Any note placed on the line or touching the named line has a specific pitch value, as described in Figure 3.104. Any note

Symbols

timbre









	sound distortion intensity
	air sound intensity
	“cold” sound intensity (sul pont/multiphonic)
	“warm” sound intensity (sul tasto/full air stream)
	trem/flz tremolando or flutter tongue
	air sound tremolando/flutter tongue intensity
	vib vibrato intensity and length
	short sharp sound

Figure 3.103: *Socket Dialogues*, timbre notation

placed above or below, not touching the line, has an indeterminate pitch value and should be freely interpreted by the performer based on the relative distance from the pitch line.

Pitch line rules




D^b —●— (D flat)	note on the line: play named pitch
D^b —  ●— (C)	note on the line with a pitch modifier: play semitone below/above the named pitch
D^b —  ●— (D flat)	note on the line with a natural modifier: play named pitch
D^b —●— (E flat)	note touching the line: play whole tone above/below the named pitch
D^b —  ●— (E)	note touching the line with a pitch modifier: play semitone below/above the note touching the line
D^b —●—	note not on the line: free pitch selection distance from the line indicates approximate size of the interval from the named pitch

Figure 3.104: *Socket Dialogues*, pitch line notation rules

The accidentals used in *Socket Dialogues* are modified versions of the traditional symbols, indicating a different function when applied to the trailing note. The pitch line name, however, may contain a traditional accidental symbol indicating the transposed instrument pitch (e.g. D^b). *Socket Dialogues* supports only the semitone accidentals. The score is written with the concert pitch line names. The transposition strategy is then used to display the correct pitch line names to each musician based on the selected transposition. Figure 3.105 illustrates the application of the named pitch notation in the *Melody* dialogue.

The same page transposed for B^b instruments is shown in Figure 3.106.

Figure 3.105: *Socket Dialogues*, named pitch line notation example in concert pitch

Figure 3.106: *Socket Dialogues*, named pitch notation transposed to B \flat

3.6.2.3 Transposition Strategy

The transposition strategy contains the logic required to extract named pitch information from the score and calculate appropriate display values for each connected client. From the web score perspective, transposition is effectively another overlay that modifies the source notation. As with all other strategies, the transposition configuration is stored in the `strategyConfig.yml` file on the server.

```
transpositionStrategy:
  isActive: true
  topStaveYRef: 114
  topStaveXRef: 80
  ...
  pages:
    - { pageNo: 1, part: Present, textElements: [
      { dx: 0, dy: 114.83, txt: G }, { dx: 0, dy: 104.32, txt: C }
    ]}
    ...
```

The configuration contains stave layout spatial reference data required to determine coordinates for named pitch positions and the list of all named pitches in the score, sorted by page and part. The named pitch information contains its relative x and y coordinates and the concert pitch text value. For convenience, the Adobe Illustrator JavaScript score export plugin was enhanced

to produce a list of all named pitches in the score. Figure 3.107 shows the Illustrator export plugin dialog with the new “Export NoteInfo” option. When enabled, this option creates a file with an appropriately formatted list of named pitches per page, ready for the insertion into the configuration file.



Figure 3.107: Enhanced Adobe Illustrator score export plugin

The score export plugin relies on the specific naming convention of Illustrator text elements containing a named pitch value. The allowed pitch name list is defined as: [“A”, “B”, “C”, “D”, “E”, “F”, “G”]. The only allowed pitch name modifiers in *Socket Dialogues* are semitone indicators “F” for flat and “S” for sharp. For example, “EF” stands for E flat and “DS” for D sharp. An example of pitch text element naming in the Adobe Illustrator layer window is shown in Figure 3.108.



Figure 3.108: Named pitch implementation in Adobe Illustrator

3.6.2.4 Score Builder Strategy

The Score Builder contains and manages score building blocks such as sections or movements. Each section is defined by the page range indicating its start and end point. The section information also contains dynamic mapping between connected web clients and assigned instrument parts and the section owner id. In *Socket Dialogues* the section owner is equivalent to the Presenter whilst instrument part ids represent available dialogue roles. As with all other strategies, the configuration is stored in the `strategyConfig.yml` file. *Socket Dialogues*’ configuration example contains the following parameters:

```

builderStrategy:
  isActive: true
  sections: [ pitch, rhythm, melody, timbre, impro ]
  assignmentType: USER_ANY_FCFS
  isStopOnSectionEnd: true
  pageRanges:
  - range: { start: 1, end: 12 }
    name: pitch
    instruments: [ all ]
  ...

```

The “sections” parameter lists active sections used in the particular performance. The “assignmentType” parameter determines the mechanics of the section owner assignment. The value “USER_ANY_FCFS” indicates that the First Come First Served algorithm should be used for the section Presenter assignment, which, in *Socket Dialogues*, is the equivalent to the section owner. Other possible section owner assignment types are defined in the enumeration `SectionAssignmentType` as:

```

public enum SectionAssignmentType {
    MANUAL, AUTO, USER_ANY_FCFS,
}

```

The configuration then specifies the page range for each section. In the example above, the “pitch” section’s start page is 1, and the end page is 12. The parameter “isStopOnSectionEnd” specifies whether the play should be stopped automatically when the section ends or continue to the next section.

The Score Builder strategy allows for the independent management of score units that can be played in any order. As large Adobe Illustrator files tend to become cumbersome and too slow for efficient score authoring, the *Socket Dialogues* score was split into separate files, one for each section. The software utility Score Merger, written in Java, was created to merge separate files into a single score formatted for ZScore software. The combination of the Score Merger and the Score Builder can be utilised to manage scores of any size and complexity.

3.6.2.5 Audience Feedback Loop

Socket Dialogues’ audience score representation provides a voting mechanism during the dialogues’ play. The intention behind this mechanism was to include the audience in the concept of a dialogue (agree/disagree) and, furthermore, to provide real-time feedback from audience members to all other performance participants about the current state of play. Inspired by the social media approval icons, the audience view deploys a simple layout with the thumbs up and down icons on each side of the vote-metre showing the aggregate vote count (Figure 3.109). The vote count is a simple server-side algorithm, adding or subtracting 1 from the total vote

for each thumb up or down click, respectively. The audience view vote-metre reflects the vote count and the momentum direction: going up into the green, indicating a more positive vote, or down towards the red, indicating a more negative vote. The aggregate vote value is scaled to the numerical range [1, 10] in both directions for visualisation and audio representation purposes.

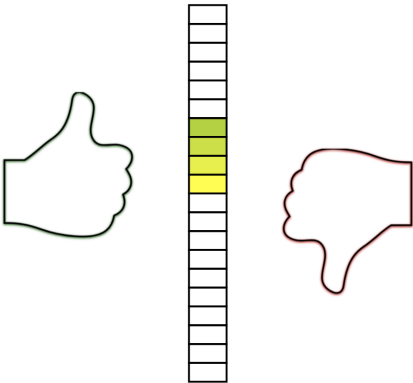


Figure 3.109: *Socket Dialogues*, voting audience view

A representation of the *Socket Dialogues* audience vote-metre, showing the current vote count, is available to all participants. The musicians’ web score view shows the vote-metre in the top right corner (Figure 3.110).

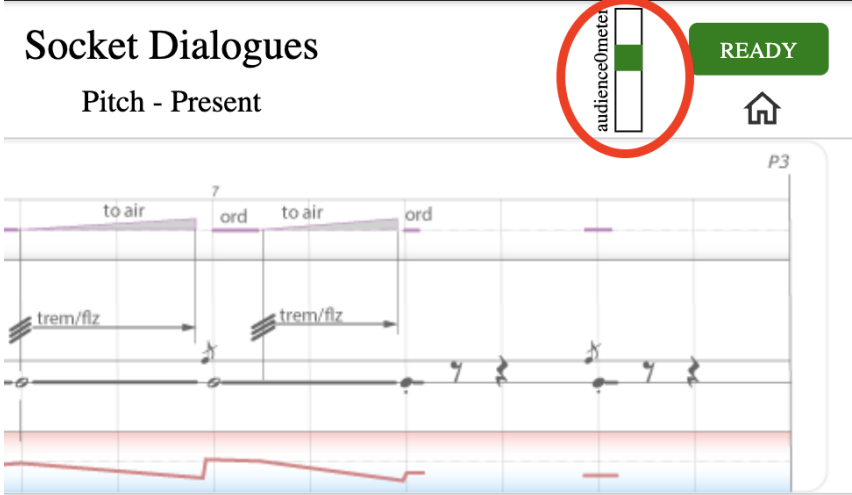


Figure 3.110: *Socket Dialogues*, Web Score audience vote view

The Control GUI audience vote-metre representation, used by the composer/conductor, is placed in the middle of the *Socket Dialogues* tab (Figure 3.111).

In addition to the visual vote count representation, the audience score also produces audio output linked to the vote sign and momentum direction. The implementation of the ZScoreMeter JavaScript object ties audience mobile device audio gain to the absolute vote count value and

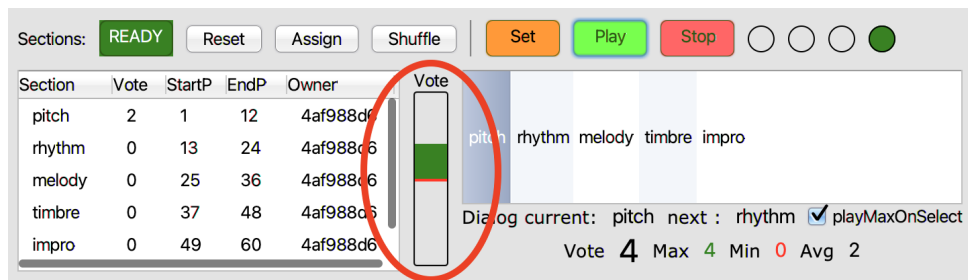


Figure 3.111: *Socket Dialogues*, Control GUI audience vote view

changes the type of the audience mobile device audio output depending on the vote sign. If the vote is positive, audience mobile devices play audio scheduled by the *Socket Dialogues* score. However, if the vote is negative, the mobile phone devices start producing filtered white noise that gets louder the higher the negative vote is. The white noise generator (zsNoise.js) uses a set of frequencies sent from the server to tune its internal Biquad bandpass filter and Q (quality factor) value as required by the currently played dialogue. The Q value is inversely proportional to the vote count, making the white noise more prominent for the higher negative vote and more pitched for the lower the negative vote.

The conductor and musicians can choose to ignore the audience vote or react with an ad hoc or a pre-agreed choice of options, ranging from the gentle modification of the performance character to a drastic change of the played material content. The audience can then react to the musicians' and conductor's responses, thereby closing the dynamic feedback loop between all participants. This concept creates novel relationships between different participant types that can lead to unforeseen outcomes. The heightened awareness of the fellow performance participants and their participation agency inevitably impacts, for better or worse, the character of the performance. This may require adjustments of embodied sensibilities and the development of new awareness skills on behalf of musicians used to one-way communication flow.

3.6.2.6 Mobile Instrument

In *Socket Dialogues*, the audience can be given the agency to produce a pitched sound notated in a simplified symbolic format directly from their touch devices. The notation representation replaces thumbs up and down view with a selection of conventional symbols representing sound duration, as illustrated in Figure 3.112. The stem directions of the notes mimic thumb positions displayed in the voting view (Figure 3.109). The available musical note durations include minim, crotchet, quaver, and semiquaver. The actual sound duration is calculated from the current tempo and time multipliers received from the Control GUI or the server-side configuration.

The pitch of each note is determined through a random selection from a frequency array specific to the currently played dialogue. Frequencies associated with the stem up note are always in the higher spectrum compared to the stem down note. The audio is generated in zsSynth.js by two sine oscillators set to the same frequency but employing different detune and

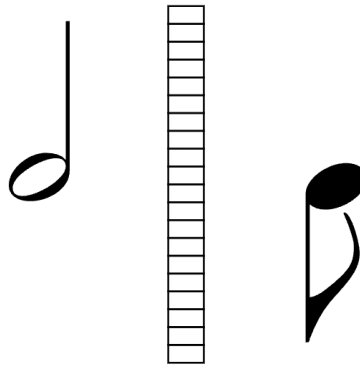


Figure 3.112: *Socket Dialogues*, voting audience view

frequency modulators. The gradual detuning between the two oscillator frequencies produces beating and chorus effects. A combination of different mobile devices used by the audience creates a spatial distribution of unstable synthetic sounds centred around the selected frequency pool.

The notation graphic is defined in the HTML file used by the audience as a set of SVG symbols referenced from JavaScript. For instance, the symbol for a crotchet with the stem up is defined as a path derived from the open-source LilyPond notation library:

```
<symbol id="crotchetUp" data-name="crotchetUp">
  <path id="crotchetUpPath" pointer-events="none"
    d="m 13,2 0,31.04 c -0.84,-0.84 -2.08,-1.28 -3.68,-1.28 -4,0
      -9.12,3.32 -9.12,7 0,2.2 1.92,3.44 4.48,3.44 4.2,0 9.2,-3.4
      9.2,-6.96 L 13.88,2 13,2 Z" />
</symbol>
```

When one of the note symbols is touched, the audience score representation generates a sound, as described above, and initiates a GSAP tween animation that spans the entire duration of the note. The tween performs a randomised movement of the selected note within the given spatial boundaries, gradually reducing its size and transparency. When a note is triggered, the audio rendering functionality is disabled for both notes until the GSAP tween is completed. This strategy ensures that each mobile device is effectively a monophonic instrument and mitigates excessive note triggering by the audience.

3.6.3 Score Commentary

Socket Dialogues' score consists of a number of musical dialogues that act as individual movements within the larger structure. As illustrated in Figure 3.96, twelve aspects of music were mapped to the circle of fourths, out of which five dialogues – pitch, rhythm, melody, timbre, and improv – were implemented for the purpose of this research. The inspiration for each dialogue's

structure was taken from Kennedy’s (2010) analysis of Plato’s literary works and the definition of the harmonious ratios, as displayed in Figure 3.95. Following previous tests, the overall tempo curve was limited to the most playable range of 60 - 100 bpm, with the main anchor points set at 60, 80, and 100 bpm. The order of play is dynamically selected by the performers at the beginning of a performance. The composer/conductor can modify the order of dialogues or their tempo, if required, from the Control GUI. In addition to notated dialogues, a free improvisation interlude driven by the audience can be performed before or after any dialogue. *Interludes* include all performance participants in a direct and open musical exchange.

3.6.3.1 Pitch

Pitch is a subjective perception of an objectively measurable sound wave frequency. This dialogue explores the relationships between acoustic interpretations of a notated pitch, digitally generated random noise, and its transformation into a detectable frequency through audio filtering.

Kennedy’s discovery of the significance of the number twelve in the structure of Plato’s dialogues (2010) evoked various subjective responses related to the musical pitch and compositional structuring. To start with, all dialogues were split into twelve sections, each consisting of twelve beats. The immediate subjective response to this structural division was a memory recall of a simple twelve-bar blues form (Figure 3.113). The functional harmony chart of the twelve-bar blues, simplified into pitch intervals, provided the basis for the compositional development. The tonal centre was set to G based on the mapping in Figure 3.96, whilst the initial tempo was set to 80 bpm, which lies in the middle of the *Socket Dialogues*’ tempo range. The notions of a section, page, and bar are equivalent, having the same durations throughout *Socket Dialogues*.

pitch	bar	1	2	3	4	5	6	7	8	9	10	11	12
	Blues, G, 80												

key:

I	IV	V
---	----	---

Figure 3.113: *Socket Dialogues*, *Pitch* dialogue structure

The opening statement of the dialogue (Figure 3.114) exposes the tonal centre in the lower end of the instrument range, played quietly with the timbre notation asking for the “air” start, a barely audible sound that requires different interpretation on different instruments. Accents on beats 8, 9, and 12 correspond to the “harmonious” intervals in Kennedy’s analysis of Plato’s dialogues (Figure 3.95). In this instance, the meaning of an interval is applied to time. As each bar consists of twelve beats, the “harmonious” intervals simply refer to the particular beat onsets within the bar.

The prominent feature permeating all dialogues is the use of an acciaccatura before accented notes. The inspiration for this kind of acciaccatura combined with a long drone comes from the

traditional Ganga singing indigenous to rural Bosnia and Herzegovina (Crnačka Ganga 2014).

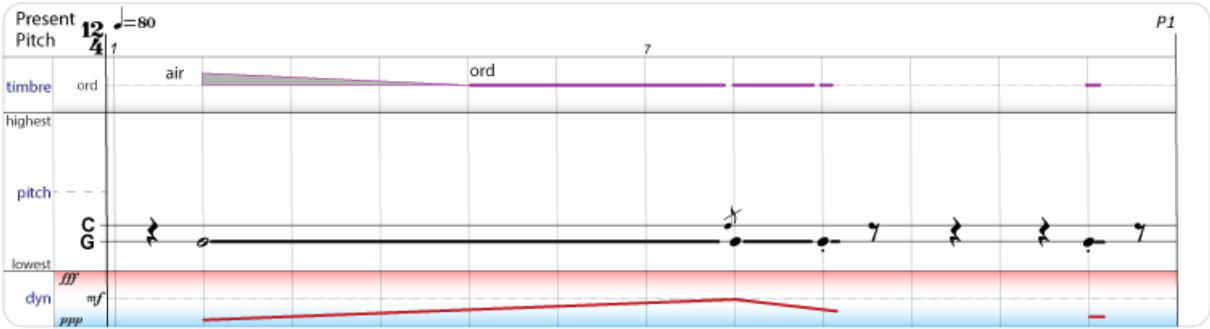


Figure 3.114: *Socket Dialogues, Pitch*, opening statement

In the *Pitch* dialogue, the acciaccatura is played a perfect fourth above the held note. Based on Kennedy’s interval “harmoniousness” (Figure 3.95), it is possible to construct a chromatic scale starting from the tonal centre, as illustrated in Figure 3.115. This table is used as a guide for the interval relationships in the dialogue.

1	2	3	4	5	6	7	8	9	10	11	12	Neutral
G	Ab	A	Bb	B	C	Db	D	Eb	E	F	Gb	Harmonious
												Disharmonious

Figure 3.115: *Socket Dialogues, Pitch*, interval “harmoniousness”

Whilst the Presenter plays the dialogue exposition, all other musicians are instructed to listen and select the role they wish to perform (Figure 3.116). The available options are: Agree, Disagree, or Abstain. The role selection effectively maps the musician’s view to the notation specific for the role. The selected role can be changed during the dialogue at specific time windows.

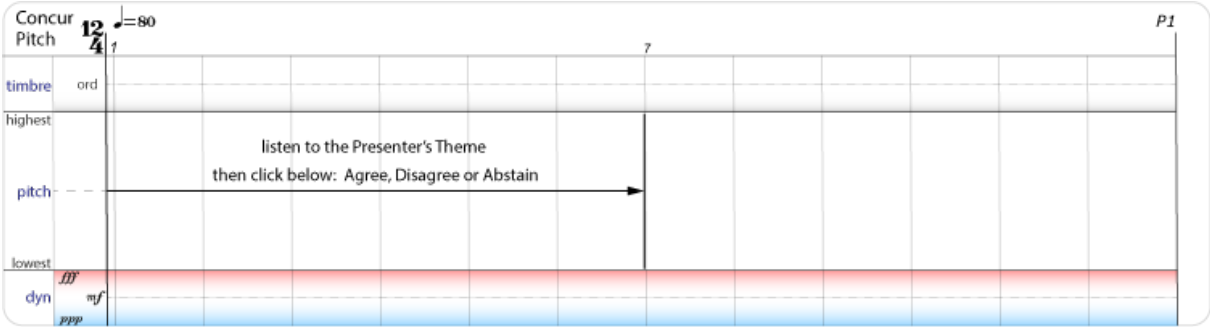


Figure 3.116: *Socket Dialogues*, opening page for all musicians who are not the Presenter

The Concur part echoes the material played by the Presenter on the previous page, whilst the Dissent part uses a modified version transposed by the minor second, which is considered a dissonant interval in Western classical music. The Dissent reply also contains the augmented fourth (D^b), known as the “the devil’s interval”, which was banned in Renaissance church music and is also classified as “disharmonious” in Kennedy’s analysis (Figure 3.117).

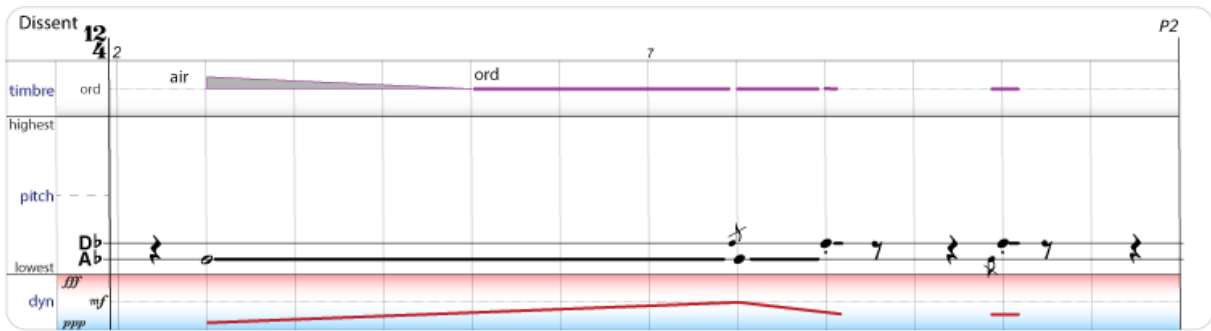


Figure 3.117: *Socket Dialogues, Pitch*, Dissent part reaction to the Presenter's statement

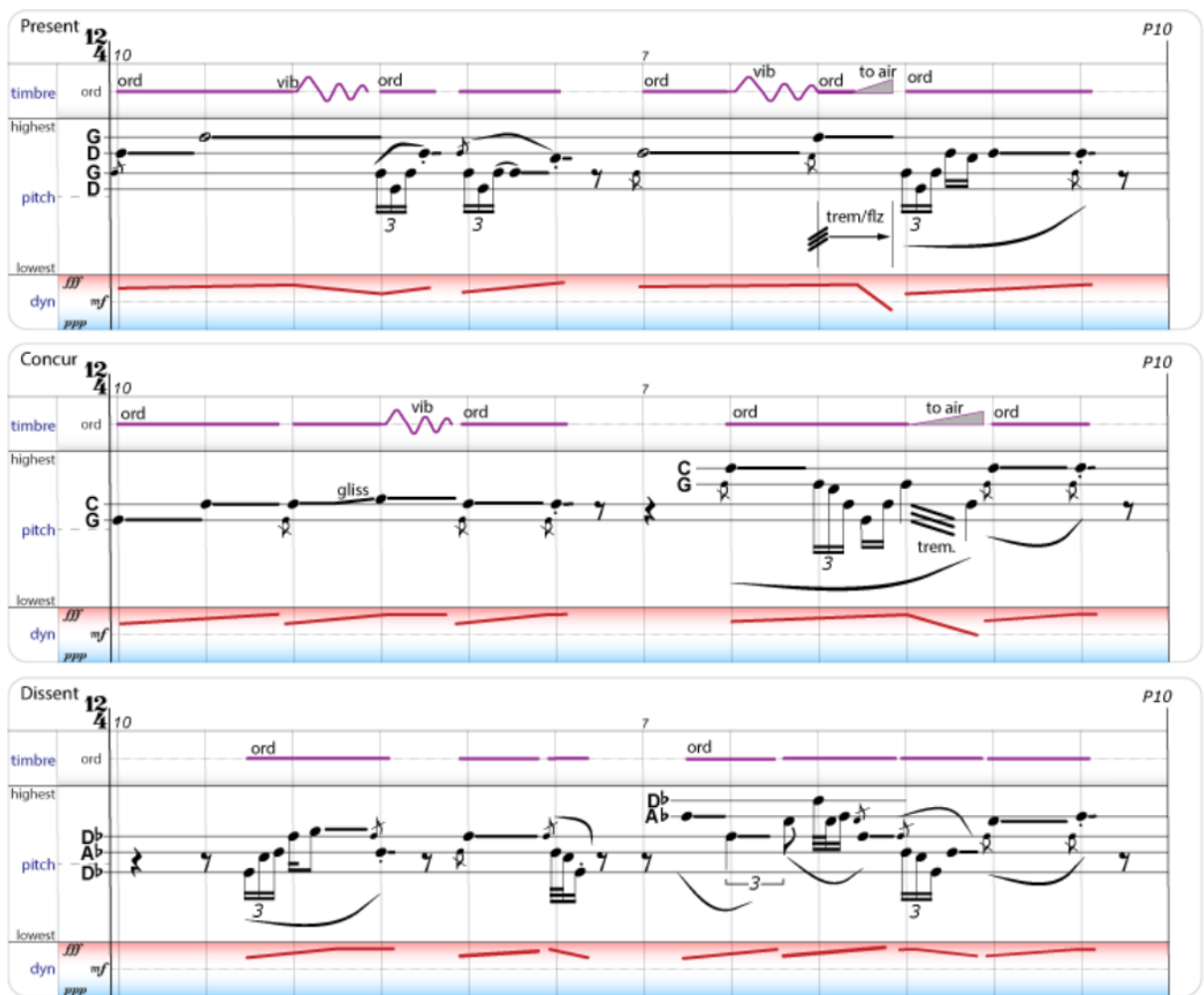


Figure 3.118: *Socket Dialogues, Pitch*, bar 10 notation

The dynamic and textural peak of the *Pitch* dialogue is reached in bars 9 and 10 as expected by the twelve-bar blues form (Figure 3.113). The complexity of the notation gradually increases to reach four named pitch lines in bar 10 (Figure 3.118). All instruments play flurries consisting of perfect fourth and fifth intervals embellished with acciaccaturas introduced in the opening statement. Whilst the Present part focuses on the perfect fifth (D) of the tonal centre (G), the Concur part plays the perfect fourth (C) in alignment with the blues form. In contrast, the

Dissent part fixes on the minor second (A^b) and augmented fourth (D^b) with a passing major seventh (G^b) and augmented fifth (E^b).

The Present and Concur parts end the dialogue with the recapitulation of the opening statements in a slightly modified form that indicate agreement, whilst the Dissent part ends with the unsettled rhythmical figure on beat 11 (Figure 3.119).

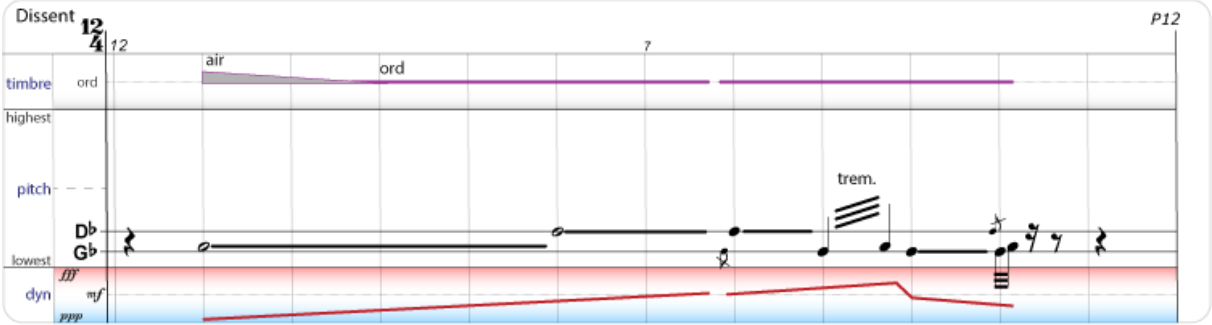


Figure 3.119: *Socket Dialogues, Pitch, Dissent, bar 12 notation*

Throughout the dialogue, the audience participates by expressing their opinion through the voting mechanism illustrated in Figure 3.109. If the vote is positive, audience mobile phones play audio samples extracted from the score authored in the Sibelius notation software, as illustrated in Figures 3.120, 3.121, and 3.122. Audio samples used by audience mobile devices were created by exporting each part in an mp3 audio file format directly from Sibelius.

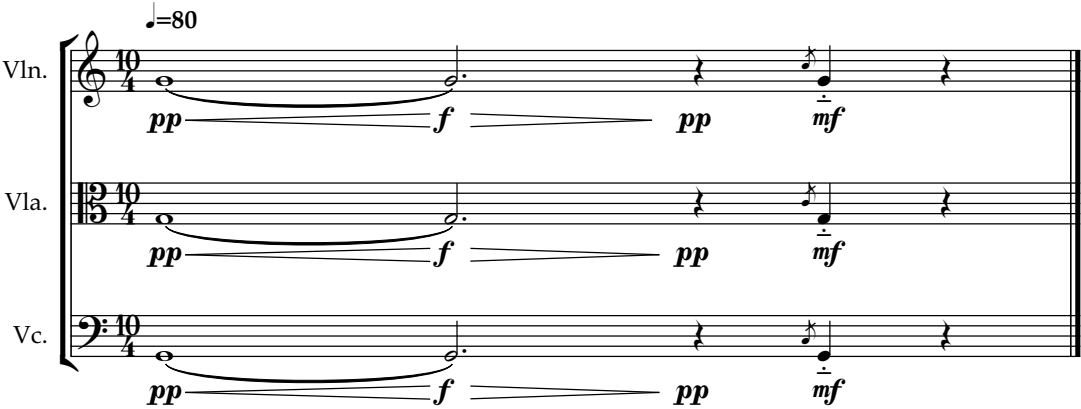


Figure 3.120: *Socket Dialogues, Pitch, Audience audio score, tonal centre*

All the audience audio score bars above are only ten beats long and end with a rest. The shorter bar lengths allow for more flexible scheduling of the audience audio and compensate for possible Wi-Fi latency. The rests at the end of the bar ensure that the exported audio ends at the zero signal level, thus avoiding any unnecessary clicks on audience mobile devices. All exported audio files were normalised to -1 dB peak for maximum effect, as their output volume is ultimately decided by the audience vote, as described above (Figure 3.109).

The audio export process from Sibelius produced nine files, one for each of the three parts (vln, vla, and vc), and each pitch value (tonic, perfect fourth, and fifth). The file names

Figure 3.121: *Socket Dialogues, Pitch*, Audience audio score, perfect fourth

Figure 3.122: *Socket Dialogues, Pitch*, Audience audio score, perfect fifth

were added to the configuration file `audienceScoreConfig.yml` under the preset id 1002 (e.g. “`/audio/DialogsPitch1-1.mp3`”) as:

```
- id: 1002
  config:
    - player: {
      audioFiles: [
        "/audio/DialogsPitch1-1.mp3",
        "/audio/DialogsPitch1-2.mp3",
        "/audio/DialogsPitch1-3.mp3",
        "/audio/DialogsPitch2-1.mp3",
        "/audio/DialogsPitch2-2.mp3",
        "/audio/DialogsPitch2-3.mp3",
        "/audio/DialogsPitch3-1.mp3",
        "/audio/DialogsPitch3-2.mp3",
        "/audio/DialogsPitch3-3.mp3"
      ],
```

```

    audioFilesIndexMap: [
      [],
      [0, 1, 2],
      [3, 4, 5],
      [6, 7, 8]
    ],
  }
  ...

```

The preset configuration above also contains the “audioFilesIndexMap” parameter, containing index pointers to the “audioFiles” array. The first populated member of the “audioFilesIndexMap” containing indexes [0, 1, 2] refers to the tonic sample files, the second nested element ([3, 4, 5]) refers to the perfect fourth files, whilst the last element ([6, 7, 8]) refers to the perfect fifth samples. The appropriate file group is referenced in the score when required. For example, the following script executed in the *Pitch* dialogue bar 2 references the “audioFilesIndexMap” element with the index value of 1:

```
web:beat=2:webScore.setAction('play', 'AUDIO', ['player'], {'index':1});
```

This index translates to the array [0, 1, 2]. In the final step of the selection algorithm, one of the index values from the array [0, 1, 2] is randomly selected, and the corresponding file is played by the mobile device. This means that each audience member’s mobile device will play one of the following three files on beat 2 of bar 2: “/audio/DialogsPitch1-1.mp3”, “/audio/DialogsPitch1-2.mp3”, or “/audio/DialogsPitch1-3.mp3”. These files are audio exports of the Violin, Viola, and Cello parts shown in Figure 3.120. The result of such a selection algorithm is a fluctuating three-part sound texture produced by the audience mobile devices when the audience vote is positive.

If the vote is negative, audience mobile devices generate filtered white noise that progressively becomes louder and less recognisable in pitch content as the negative vote increases. The configuration for the white noise filtering is also a part of the audience score configuration preset id 1002 discussed above. It can be found under the “synth” configuration as:

```

- synth: { bpm: 80,
  durMultiplier: 8.0,
  freqMultiplier: 1.0,
  osc1Freq: [ 783.99, 946.36, 1108.73 ],
  osc2Freq: [ 392.00, 473.19, 554.37 ],
  osc3Freq: [ 392.00, 554.37, 783.99, 1108.73, 1567.98, 2217.46,
  3135.96, 4434.92, 6271.93, 8869.84 ]
}

```

The white noise bandpass filter frequencies used in the *Pitch* dialogue are specified under the “osc3Freq” configuration. The “osc1Freq” and “osc2Freq” frequency arrays relate to the

notes with stems up and down, respectively (Figure 3.112). The configured filter frequencies (“osc3Freq”) correspond to pitches G and C# in different octaves, ranging from G4 (scientific designation, where C4 is the middle C) to C#9. The choice of the augmented fourth interval aligns with the compositional material used in the Dissent part.

The algorithm responsible for determining the actual bandpass frequency and Q value on each audience member’s mobile device first takes a chunk (subarray) of the configured frequencies based on the current audience vote value. The absolute vote value is scaled to the range [1,10] for easier visualisation and mathematical operability. The subarray can have up to four different frequencies, and its starting point shifts with the size of the negative audience vote. For example, if the audience negative vote is low (e.g. -1), then the frequency subarray starts from the first element (e.g. [392.00, 554.37, 783.99]). The subarray’s starting point moves up as the negative vote count increases. As the scaled vote value approaches the maximum allowed value (10), the frequency subset consists of the last four elements (e.g., [3135.96, 4434.92, 6271.93, 8869.84]).

The actual frequency used to tune the bandpass filter is randomly selected from the calculated frequency subarray. The intention behind the frequency array chunking and randomisation algorithm is to create a dynamic polyphonic sound texture across the audience mobile devices.

The Q value sent to the bandpass filter also depends on the size of the vote. It is selected from the configured value array, which consists of ten elements: [300, 100, 70, 50, 30, 20, 15, 10, 5, 3]. A higher Q value results in a narrower bandwidth profile, producing a more accurate pitched sound, whilst a lower Q value generates a wider bandwidth noise with only a hint of a pitched frequency. As the audience vote count is scaled to the range [1, 10], it can be directly mapped to the corresponding index in the configured array. For example, if the scaled vote value is 1, the algorithm selects the Q value of 300 from the array, and if the scaled vote value is 10, the algorithm selects the Q value of 3. This mapping ensures that the audience’s voting behaviour directly influences the characteristics of the filtered white noise generated on their mobile devices.

The audience score configuration preset id 1002, which was discussed above, is loaded implicitly in the first bar of the AV part using the following script:

```
web:beat=-1:reset=only:webScore.reset(2);
```

The script is classified as “reset only” because each dialogue is performed as a separate movement, triggering the reset scripts on load. The preset id 2, referenced in the script above, is defined in the audienceScoreConfig.yml file as:

```
- id: 2
  scripts: [
    "webScore.reset(1);",
    "webScore.reset(1002);",
    "webScore.setSection('pitch');",
    "webScore.activateViews('metre');",
```

```

    "webScore.deactivateViews('notes', 'audio', 'thumbs', 'vote');"
  ]

```

Amongst other calls, these scripts load the preset id 1002, initialise the audience vote metre, and deactivate all other views ('notes', 'audio', 'thumbs', 'vote'). The concept of the “view”, related to the implemented Model-View-Controller software design pattern (Krasner and Pope 1988), enables modular construction and the control of the audience score representation. Each “view” can be instantiated either from the score script or from the Control GUI Dialogues tab (Figure 3.123). Any enabled modular “view” is instantly activated on the audience’s mobile devices. For example, the combination of the “metre” and “thumbs” views generates the graphics shown in Figure 3.109.

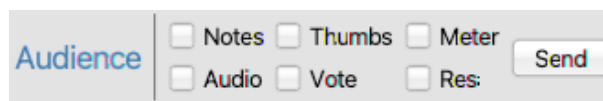


Figure 3.123: *Socket Dialogues*, Control GUI, audience view controls

Similar to other scores in the portfolio, *Socket Dialogues* triggers audio files loaded in ZScore’s Max patch buffer players. Dialogues use a number of audio files, each associated with a particular score page. For example, the script that loads file “DialogsPitch_b3.wav” associated with the *Pitch* dialogue bar 3, is placed in AV part, page 2:

```
max:beat=8:setFile,b1,DialogsPitch_b3.wav
```

The digital audio files loaded into Max buffer players were first authored in Logic software and then mixed in Pro Tools. In the *Pitch* dialogue, the digital audio supports pitch frequencies played by the Presenter. Frequencies below 200 Hz and over 3000 Hz, complementary to most acoustic instruments, are emphasised, whilst the notes embellished with acciaccaturas are underpinned with additional accents.

3.6.3.2 Rhythm

This dialogue explores various layers of rhythm, encompassing a regular pulse represented by the bass drum kick, the grouping of notes into metric subdivisions and irrational tuplets, and the broader-scale periodicity or events composed of alternating blocks of sound and silence. Kennedy’s definition of “harmonious” intervals (Figure 3.95) served as the foundation for creating a series of time periods expressed in a number of beats (Figure 3.124). A symmetrical rhythmic structure was constructed from this series by implementing each period initially as a sound and then as silence.

Figure 3.125 demonstrates the implementation of the first three time periods from the series. The Presenter’s exposition begins with a single note, followed by a crotchet rest, and proceeds with groups of two and three beats of alternating sound and silence. In the subsequent bars, the

rhythm																
C, 100	ratios	1	2	3	4	6	8	9	12	9	8	6	4	3	2	1

Figure 3.124: *Socket Dialogues*, *Rhythm* block durations in number of beats

durations of the event periods and the complexity of notation gradually increase in accordance with the series depicted in Figure 3.124, culminating in bars 6 and 7 with the block of sound lasting twelve beats. From then on, the event period durations reverse back to the starting point in a retrograde series of events, ending with a single note on the last beat.

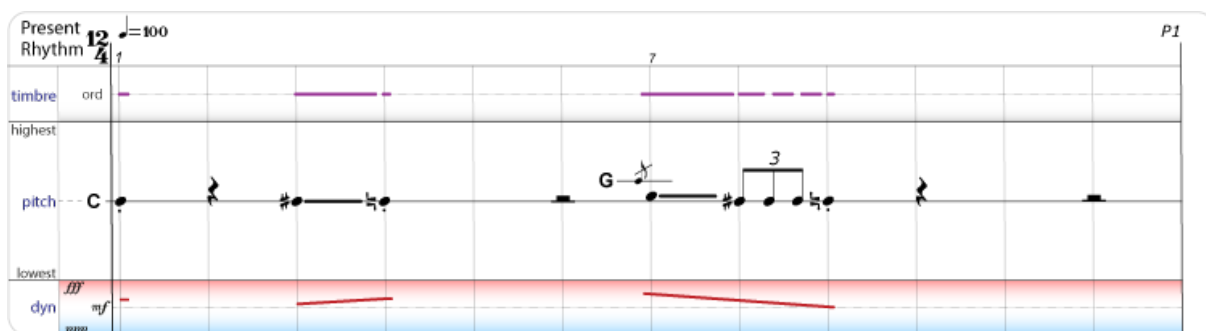


Figure 3.125: *Socket Dialogues*, *Rhythm*, Present, bar 1 notation

Sound blocks consist of groups of notes organised into phrases whose sizes adhere to the same numerical series as the rhythmic structure (Figure 3.124). The phrasing in longer blocks employs guided indeterminate notation, where the durations of individual notes are precisely notated, whilst their pitches are approximated based on the note's relative vertical distance from adjacent notes and the tonal centre (C), as illustrated in Figure 3.126.

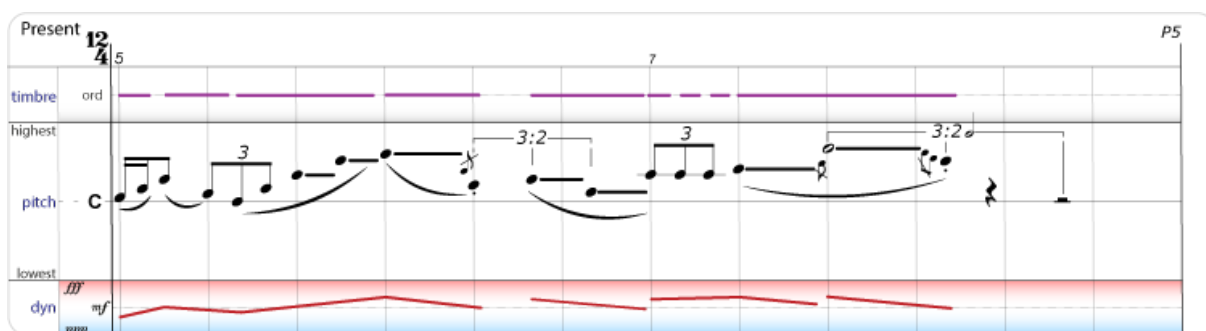


Figure 3.126: *Socket Dialogues*, *Rhythm*, Present, bar 5 notation

The Concur part responds by mirroring the Presenter's block structure and note groupings, offset by one bar length, until they meet on page 6 to play the longest twelve-beat block together. Following the dialogue climax on pages 6 and 7, the Concur part assumes the lead by quoting the remaining sound blocks ahead of the Presenter until they converge in the last bar, ending on the single note together. The Dissent part entries align with the timing of the Concur sound blocks, however, the grouping of notes is limited to irrational odd tuplets (e.g. 5:4, 7:4, 3:2).

This approach creates contrasting rhythmic phrasing patterns whilst ensuring the integrity of block timings when different role combinations are selected during a performance.

The bass drum kick rendered by the Max patch delivers the fundamental pulse, acting as the sound block beat counter. In combination with high-hat and snare samples, it also evokes the sound texture reminiscent of the ubiquitous contemporary electronic dance music. The association between rhythm, movement, and the bass drum has been a prominent feature of electronic dance music culture since the late 1980's (Dyck et al. 2013, p. 17). The *Rhythm* dialogue builds on this cultural link by extending the traditional 2 and 4 beat patterns of dance music into blocks of varying lengths and layering the rational and irrational tuplet phrasing on top of the basic pulse.

In the case of a positive audience vote, the audio produced by audience mobile devices consists of high frequency percussive sounds, such as high-hat and various cymbals. The timing of the sounds complements the Presenter's rhythmic patterns. Due to the intermittent Wi-Fi latency and varying quality of mobile device speakers, the impact of the audience audio is comparable to multi-channel delay and chorus effects. This adds sonic complexity whilst essentially echoing the Presenter's theme. The negative audience vote triggers a noise pattern similar to the *Pitch* dialogue, except that the filter frequencies are set to the dialogue's tonal centre pitch (C) and the augmented fourth (F#) in various octaves:

3.6.3.3 Melody

Contemplation of the importance of the number twelve in Plato's dialogues and its potential connection to musical melody led to obvious associations with serialism and twelve-tone composition techniques. As a result, a decision was made to create a melody consisting of all twelve notes of the chromatic scale as the Presenter's opening subject (Figure 3.127).



Figure 3.127: *Socket Dialogues, Melody*, Presenter's theme

The theme begins on E^b, the predetermined tonal centre of the dialogue. The structure of the dialogue was derived from the theme using different compositional techniques such as inversion, segmentation, transposition, rhythmic augmentation and diminution, intentionally deviating from the strict constraints of serialism (Figure 3.128).

Instead of a straightforward chromatic inversion, an unusual interval inversion of the main theme was created in Sibelius by employing diatonic inversion around E^b in an unspecified key signature (Figure 3.129).

In contrast to the other dialogues, the Presenter's opening statement lasts for two bars and comprises the theme and its inversion. The main theme (Figure 3.127) translated into the named

melody	bar	1	2	3	4	5	6	7	8	9	10	11	12
Eb, 80	P	T	Tinv		Tseg1		Tseg2		Tseg	Tinv seg		T	Tinv
	O			C/R		C/R		C/R	C/R	C/R		Tinv seg	

key: **P**: presenter, **O**: other, **T**: theme, **inv**: inversion, **seg**: segment, **C/R**: call/response

Figure 3.128: *Socket Dialogues*, *Melody* dialogue structure



Figure 3.129: *Socket Dialogues*, *Melody*, Presenter's theme inversion

pitch line notation is depicted in Figure 3.105, whilst its inversion is presented in Figure 3.130.

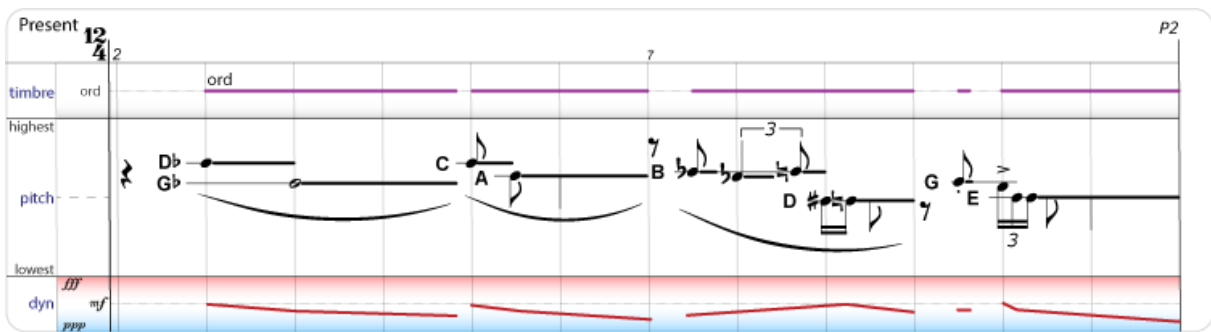


Figure 3.130: *Socket Dialogues*, *Melody*, Presenter's theme inversion in Web Score notation

Audience and Max audio behave similarly to other dialogues, with audio frequencies and textures selected to align with the dialogue's musical material.

3.6.3.4 Timbre

This dialogue invites musicians to explore the sonic possibilities of their instrument by following instructions based on generic musical terminology. The dialogue structure and the key for various technique abbreviations are presented in Figure 3.131.

timbre	bar	1	2	3	4	5	6	7	8	9	10	11	12
F#, 60		A	A/P	P/Pr	Pr/S	S/Lg	Lg/mf	Mf/L pp	L/M mf	M/H ff	H/D ff	P/Pr	Pr/A

Legend: **A**: air, **P**: pitch, **Pr**: percussive, **S**: staccato, **Lg**: legato, **Mf**: multiphonics, **D**: distorted
L: low, **M**: mid, **H**: high, **pp**: quiet, **mf**: half loud, **ff**: loud

Figure 3.131: *Socket Dialogues*, *Timbre* dialogue structure

Musicians are free to interpret the instructions in a manner they deem most suitable for their instrument. The term “air” can be interpreted as any non-pitched sound containing a degree of white noise, whilst “multiphonics” may indicate any polyphonic sound.

The Present part starts with quietly played noise and gradually builds towards the climax in bars 9 and 10, concluding with a burst of white noise. The opening of the Presenter's exposition is depicted in Figure 3.132. Rhythmic and pitched motifs borrow material from other dialogues, establishing a sense of continuity. The combination of the acciaccatura (*Pitch*) and the triplet diminution (*Rhythm*) commencing on beat 8 in Figure 3.132 recurs in different variations throughout the dialogue, acting as a unifying device that links all parts.

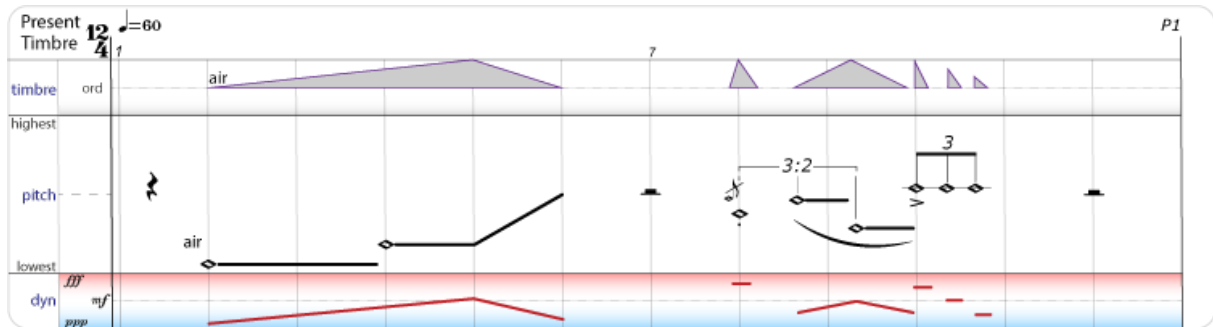


Figure 3.132: *Socket Dialogues, Timbre, Presenter's opening theme*

In line with previous dialogues, the Concur part echoes the Presenter until they meet together in bar 10 (Figure 3.133), albeit in different registers. Black triangles in the timbre notation (Figure 3.133) indicate sound distortion, whilst the blue triangle serves as a quick visual prompt to play multiphonic (polyphonic) sound.

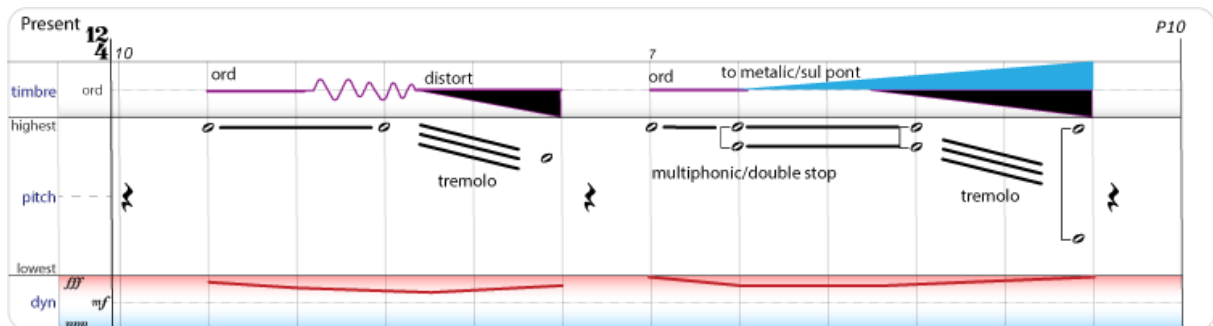


Figure 3.133: *Socket Dialogues, Timbre, Presenter's bar 10 score*

The notation for the Dissent part employs the same musical material as other parts but in an inverted structural order. As in other dialogues, audience and Max audio are designed to align with the musical material of the dialogue.

3.6.3.5 Improv

This dialogue removes even more notational constraints, granting musicians greater decision-making agency and expressive freedom. The performance instructions, employing simple musical terms and graphic notation, aim to establish a recognisable compositional structure (Figure 3.134).

improv	bar	1	2	3	4	5	6	7	8	9	10	11	12
	Subject	Ln L	Sh M mf	Ln L	Sh Sp M	Sh Dn M	Lg Dn L	Ln Sp M	Dn H ff	Dn M ff	Ln/L	Ln Sp pp	
Bb	tempo	60	70	80	70	80	100	110	100	90	80	60	40

key: **Ln**: long, **Sh**: short, **Sp**: sparse, **Dn**: dense, **Rh**: rhythmical, **Lg**: legato, **L**: low, **M**: mid, **H**: high, **pp**: quiet, **mf**: halfloud, **ff**: loud

Figure 3.134: *Socket Dialogues, Improv* dialogue structure

The Presenter has complete freedom to construct and perform a dialogue exposition of their choice (Figure 3.135), whilst the other musicians actively listen and respond.

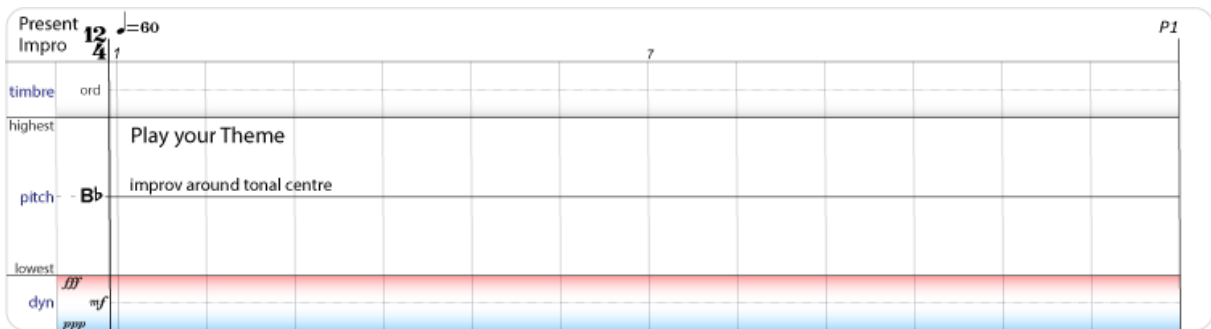


Figure 3.135: *Socket Dialogues, Improv*, Presenter's theme free improvisation

The Concur part asks players to imitate the Presenter's theme in various registers, durations, and densities, whereas the Dissent part requires a musical reaction that opposes the theme in some musical sense, through alterations in timbre, pitch, melody, or any other aspect of the Presenter's exposition. The compositional development follows the same pattern employed in the *Timbre* dialogue, culminating in bars 9 and 10 with a loud sound distortion in high registers. The movement of the tonal centre adheres to the same interval structure as the *Pitch* dialogue. The tempo curve shown in Figure 3.136 induces variations in the musician's responses, progressively building momentum ahead of the dynamic climax of the dialogue. Both the textural and temporal tensions are then gradually released towards the conclusion of the dialogue.

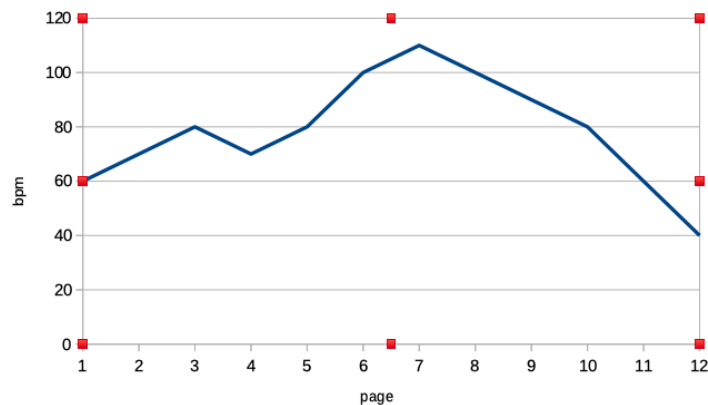


Figure 3.136: *Socket Dialogues, Improv*, tempo curve

3.6.3.6 Interlude

An *Interlude* is effectively a free improvisation dialogue where the audience assumes the role of the Presenter. It can be performed before or after any dialogue and can be repeated any number of times. The schedule and timing of *Interludes* can be agreed upon in advance with the performers or introduced by the conductor as needed.

The Control GUI features a number of presets that facilitate the initialisation of appropriate score representations for the audience and musicians, as well as the audio configuration for mobile devices and Max patch (Figure 3.137).

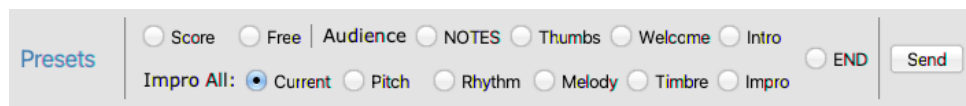


Figure 3.137: *Socket Dialogues*, Control GUI presets

The “Current” preset loads the audience instrument configuration and Max audio files for the currently played dialogue, whilst any other preset under the “Impro All” label loads dialogue-specific values. The configuration contains frequencies triggered by the two note symbols displayed in the audience score representation (Figure 3.138). For instance, the *Pitch* dialogue configuration specified in the audienceScoreConfig.yml file contains the following values:

```
- synth: { bpm: 80,  
          durMultiplier: 8.0,  
          freqMultiplier: 1.0,  
          osc1Freq: [ 783.99, 946.36, 1108.73 ],  
          osc2Freq: [ 392.00, 473.19, 554.37 ],  
          ...
```

The “osc1Freq” parameter contains frequencies played by the stem-up note, whilst the “osc2Freq” parameter represents frequencies played by the stem-down note. The frequencies assigned to the “osc1Freq” parameter are consistently one octave higher than those in the “osc2Freq” array. The first frequency in “osc2Freq” array (392 Hz) corresponds to the pitch G5, the tonal centre of the *Pitch* dialogue, whilst the last frequency (554.37 Hz) corresponds to the pitch C#6, which is an augmented fourth away from the tonal centre. The middle frequency (473.19 Hz) is derived by calculating the arithmetic mean value between the first and last value in the array. This simple calculation method produces a non-standard pitch that does not conform to the equal temperament logarithmic scale steps. In this case, the calculated frequency is slightly higher than B^b eighth sharp (472.94 Hz).

The actual frequency sent to the zsSynth.js audio library is randomly selected from the configured array. The output signal is generated by two sine oscillators set to the same frequency but employing different detune and frequency modulations. This pitch selection and production process creates an ever-evolving microtonal polyphonic texture produced by mobile devices.

Click note to play

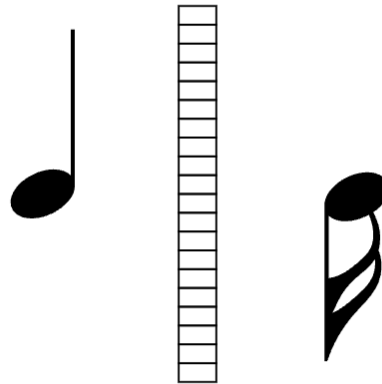


Figure 3.138: *Socket Dialogues, Interlude*, audience view

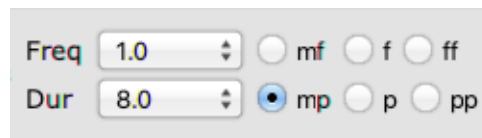


Figure 3.139: *Socket Dialogues*, Control GUI, audience instrument configuration

The duration of the triggered sound is determined by the current tempo, the selected note’s time value (e.g. crotchet), and the configured duration multiplier (“durMultiplier”). The duration multiplier parameter serves as a convenient mechanism for controlling note durations in live performance scenarios. Similarly, the frequency multiplier is used to modify the values stored in the configured frequency arrays (“osc1Freq” and “osc2Freq”). The Control GUI offers several presets that automatically adjust note duration, frequency multipliers, and the volume level of the mobile phone output. The presets are named after the dynamic markings they represent (Figure 3.139), ranging from the quietest to the loudest: “pp”, “p”, “mp”, “mf”, “f”, and “ff”. The specific values associated with each preset are illustrated in Figure 3.140.

Preset	pp	p	mp	mf	f	ff
Duration multiplier	8	8	8	2	0.5	0.05
Frequency multiplier	0.6	0.8	1	2	4	8
Gain [0,100]	10	20	30	40	50	60

Figure 3.140: *Socket Dialogues*, audience instrument preset values

In the quieter presets (“pp” to “mp”), the note duration remains constant, but it rapidly decreases in the louder presets (“mf” to “ff”), resulting in extremely short and sharp sounds for

the loudest dynamics. The frequency multiplier for the default “mp” preset (1.0) produces an unmodified pitch, whilst it decreases in the quieter presets (“pp” and “p”), resulting in lower pitches, and increases in louder presets (“mf” to “ff”), resulting in higher-pitched sound. The presets are selected by the composer/conductor in real-time during a performance.

When the free improvisation notation preset is selected (Figure 3.137), the musicians’ score views are updated with textual information prompting musicians to start free improvisation based on the audience-generated sound, as illustrated in Figure 3.141.

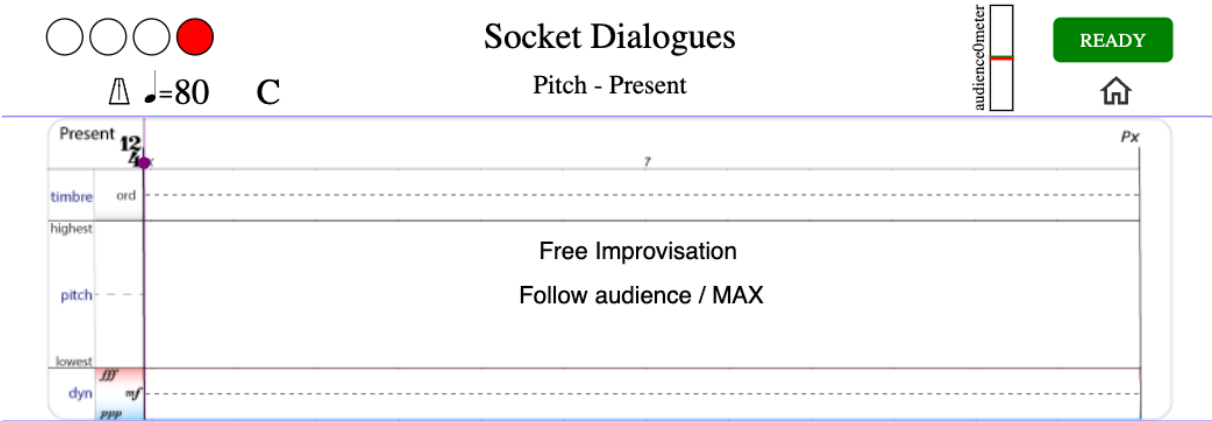


Figure 3.141: *Socket Dialogues, Interlude* web score

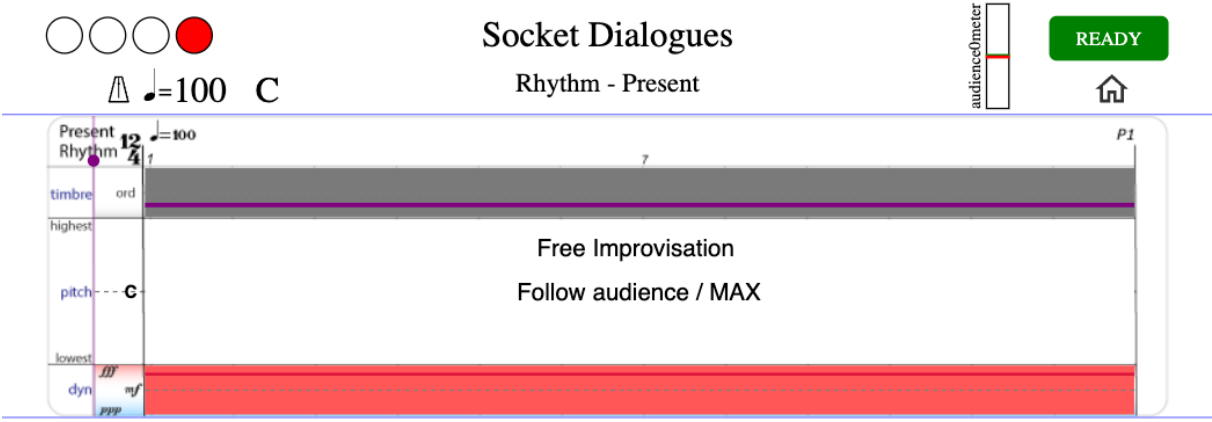


Figure 3.142: *Socket Dialogues, Interlude* web score overlays

The free improvisation notation preset (Figure 3.137) also loads dialogue-specific audio files into ZScore Max Granulator and MC Groove objects. At that point, the conductor/composer can join in as a performer using Max patch directly or through the connected hardware controllers. Additionally, the conductor/composer can change audience instrument presets, triggering a change of the audience audio texture which, in turn, acts as a modulator of the free improvisation that now involves all performance participants.

If deemed necessary, the conductor/composer can also deploy notation overlays on musicians’ notation views, as depicted in Figure 3.142. In this example, musicians are instructed to increase the intensity of their performance by playing distorted loud sounds. These additional instructions

may reduce the spontaneity of a free improvisation, however, they also provide a mechanism for communicating performance cues, such as building up to a dynamic climax or preparing for the next dialogue.

The conductor has the ability to send a text message to the performance participants from the Control GUI at any time. The message recipient selection is flexible, allowing for the targeting of any combination of musicians' roles and the audience. An example of the message configuration is displayed in Figure 3.143, where a text message is split into two lines for all musicians.

Figure 3.143: *Socket Dialogues*, text message controls

Once the conductor has composed the desired message and chosen the recipients, clicking the “Send” button triggers the transmission of the message to the designated individuals or groups – all instrumental parts in this instance (Figures 3.143 and 3.144).

Figure 3.144: *Socket Dialogues*, text message in Web Score

Chapter 4

Results And Reflections

Following extensive system testing in a controlled environment and acceptance testing with musicians in closed workshops, the ZScore system was successfully deployed in various live performances involving audience participation. In all scenarios, the system exhibited the expected functional behaviour and maintained a stable performance profile, with round-trip latencies for wired clients consistently below 10 milliseconds.

The system's performance testing results, provided in Appendix A, reveal that the current server implementation can support up to 1500 concurrent users without experiencing packet loss. A single Gigabit router and a set of Gigabit switches are sufficient to support up to 200 wired musicians, covering the majority of relevant use cases. An adequately equipped wireless access point with 2.4 GHz 2:2 MU-MIMO and 5 GHz 3:3 MU-MIMO can serve up to 100 audience members. Larger audiences of up to 500 people can be accommodated by simply adding a sufficient number of wireless access points to the network. For audiences exceeding 500 people, a more powerful 10 Gb router is required, whilst scenarios involving over 1000 connections would necessitate more complex industrial-scale infrastructure consisting of multiple web servers, load balancers, and a cluster of application server instances, as outlined in Appendix A. Audience sizes in ZScore workshops and live performances were limited to below 100 people, and therefore the basic system infrastructure outlined in Chapter 2 was found to be suitable and sufficient for accommodating the concurrent connections required for these events.

Each score included in the attached portfolio has been performed by professional musicians at least once. In several open workshops and performances, participants provided feedback through questionnaires that incorporated mixed quantitative and qualitative elements, specifically designed for musicians and the audience. The questionnaires, along with their corresponding results, are provided in Appendix B. Despite the relatively small sample size, the surveys and ad hoc conversations with performance participants have yielded valuable insights to guide further developments.

The questionnaire results consistently indicated high levels of satisfaction amongst the participants regarding the overall performance experience, receiving an average rating of 4.83 out of 5 from the musicians and 4.5 from the audience. There was minimal variation in this mea-

surement across different performances, with a mean deviation of 0.125. Similarly, participants expressed consistent satisfaction with the system's performance and stability, receiving a high rating of 4.67.

Following criticisms of the notation layout in previous versions (*Vexilla*), musicians expressed predominantly positive feedback about the ZScore notation layout and its dynamic behaviour in later versions (post *Vexilla*). Open-ended comments from musicians include "significant improvement" (comparing *Union Rose* to *Vexilla* notation) and "difficult but very performable" (*Union Rose*). The quantitative question regarding notation difficulty, where 1 indicated "Too easy" and 5 indicated "Impossible to perform", received an average score of 3.33, indicating a balanced difficulty level with a slight tendency towards a notational challenge.

Acoustic musicians responded positively to performing with digital audio (Max) in the portfolio pieces, with an average score of 4.17 out of 5. Subsequent performances (*Socket Dialogues*) received a slightly higher rating of 4.5 compared to *Union Rose* (4.0). Musicians' opinions regarding audience participation yielded a slightly lower average rating of 3.8. However, the later score (*Socket Dialogues*) received a higher rating of 4.0 compared to the earlier *Union Rose* score, which received a rating of 3.67. The interactive score features received a slightly above-average rating of 3.5, with *Socket Dialogues* once again receiving a higher rating of 4.0 compared to *Union Rose* (3.25). Qualitative open-ended responses from musicians regarding the interactive features included statements such as "wanted more" and "would be good to have more information about audience actions" (in the context of *Union Rose*). Satisfaction with the dynamic notation overlays scored an average of 4.17 across all surveyed performances.

The audience consistently reported a high level of engagement during performances (4.5) and expressed enthusiasm towards participating in future networked music events (4.5). However, the mobile device score implementation received a slightly lower average rating of 3.9. The *Union Rose* implementation scored higher (4.5) compared to the *Socket Dialogues* implementation (3.6). The survey question regarding whether the consequences of the audience's actions matched their expectations received an average score of 3.4, with significant variance ranging from 3.0 to 4.25 across different performances.

In the open-ended responses, audience members expressed predominantly positive opinions regarding their participation in networked performances. Examples of these responses include:

- "Enjoyed interaction and collective participation"
- "A great experience, I am happy to be involved in"
- "This was like therapy for me"
- "Would like to be in the know of future researches / development"
- "Music & expression is what interests me the most and I was very impressed and satisfied with what I was hearing"

However, there were also some negative responses related to the musical output and the connection between the audience's actions and their expectations. Examples of these responses include:

- “Better relation between input & output would be great”
- “The music decided by my different choice is quite similar, difficult to feel strong change”
- “Is it possible for the audience to have autonomy on how much they have control or the influence?”
- “Personally I feel that satisfaction comes a lot from the revealed impact of interaction”
- “I believe the sounds should tonally match the musicians”

When asked about the most satisfying aspects of the performance, audience responses clearly indicated that interactivity was the most highly regarded element of the experience, closely followed by the music itself (Figure B.3). On the other hand, the mobile score and digital audio received less mentions in the survey answers.

The findings indicate that the inclusion of interactive features in networked music systems can have a profound impact on the extent and nature of participant engagement in music performances. The testing results of the ZScore system suggest that interactive networked systems can be successfully implemented across various scenarios, spanning from small-scale experimental electro-acoustic performances to symphonic concerts and multimedia events hosted in large-scale venues. The most encouraging finding of this study is the evident enthusiasm expressed by both musicians and audiences towards the networked music performances, coupled with a strong interest in future developments in the field.

The ZScore system architecture and software implementation have proven to be reliable and resilient, providing a solid foundation for the creation of dynamic interactive scores. Nonetheless, the complexity of scores, which encompass mixed music notation, interactive graphics, extensive configuration, and coded algorithms, required a substantial commitment of time and effort for each composition. As it stands, the complexity of the score authoring process poses a significant barrier to the more widespread adoption of the system.

Due to limited rehearsal time with musicians, certain instrumental performances lacked interpretational accuracy. Given the nature of the dynamic notation in ZScore, musicians can only partially prepare in isolation by rehearsing from static score versions. To achieve better results, more time and effort would be needed in ensemble rehearsals with the ZScore system, allowing for a closer examination of the intricacies of the notated material and dynamic overlays. Ultimately, the only way to fully explore the possibilities of interactive music-making and improve the quality of outcomes would involve performing together with the audience in as many diverse settings as possible.

From a personal perspective, the most rewarding aspect of the project was witnessing several truly special moments during performances, where musicians, audiences, the composer, and technology collaborated to create an aural and visual experience that simply could not have been possible before. These moments included various Interludes in *Socket Dialogues*, as well as the merging of the audience’s mobile device choir, Max granulated digital utterances, and the string quartet pizzicato passages in the *I Want* section of *Union Rose*. Conversely, the fragmentation of tonal and atonal material in *Vexilla* failed to convey a coherent musical aesthetic, despite its

strong conceptual fundamentals and clear compositional intentions. A more consistent overall compositional trajectory (Gestalt) was implemented in other portfolio compositions.

The most challenging aspect of the composition process for interactive performances was finding an appropriate balance between user interface usability, meaningful audience engagement, and dynamic notation complexity, whilst maintaining creative compositional integrity. More complex user interfaces and engagement logic led to lower comprehension of musical intentions by audience members, particularly when the outcomes of users' actions were subtle and delayed. This issue became apparent in the audience feedback for *Union Rose*, where audience members had the agency to select the musical material played by the musicians. The visually rich audience score representation of *Union Rose* received a high rating of 4.5, whilst satisfaction with the perceived connection between audience actions and their outcomes received a lower rating of 3.5. Despite the comprehensive implementation of the action model illustrated in Figure 1.30, some attendees failed to correlate subtle changes in the musical material with the results of their actions.

To fully develop a mental model of the action context in this case, audience members would need to memorise the musical material played in the preceding section and link it to the collective action (highest tile vote). In *Union Rose*, this task was made more complex by having eight musical queues performed by different instruments on each repetition, posing a challenge for most audience members. Musical cues associated with graphical elements can be made more obvious and memorable through simplification and repetition, providing that the outcomes match compositional intentions. Committing musical cues to memory demands a conscious effort from the audience members, so the effectiveness of such an approach would inevitably vary based on the audience's size, their familiarity with the particular music style, and willingness to participate.

Audience members benefitted from repeated engagements in networked performances. In workshops where the same score was performed more than once, audience members reported a higher quality of experience with each subsequent performance. This outcome is unsurprising, as the mental representation of the performance context, score representation, and audience's role in a performance becomes more enriched with each repeated involvement. Broader utilisation of networked systems would improve participants' understanding of the interactive environment and help establish conventions of interactive music-making.

Another important factor to consider here is the impact of participants' focus switching. Complex score representations can demand significant cognitive effort, potentially distracting from participants' ability to fully engage with the auditory aspects of the performance. In the case of *Union Rose*, the audience score representation features continuously evolving interactive elements. Some attendees reported that the rich graphic content hindered their ability to fully immerse in music. A well-designed interactive score should incorporate designated periods that facilitate focus switching between different performance elements, ensuring a balanced and immersive experience for the participants.

Socket Dialogues employed a more obvious audience interaction model with immediate action response and a simplified user interface, leading to a higher rating for action outcomes (4.25), but a lower rating for the mobile device score implementation (3.62). The gamification of the user interface that incorporated thumbs up and thumbs down icons, ubiquitous in social media applications, and music note images associated with instant sound production helped clarify the compositional context, but somewhat reduced the aesthetic impact of the score representation. The conceptually clear *Socket Dialogues* interface also helped with the focus switching, enabling the audience to transition their attention more effectively towards music listening.

The audience feedback indicates that having full control over action outcomes and the incorporation of gamified user interfaces has a positive impact on audience satisfaction ratings. On the other hand, the user interface design and interaction model should be a product of compositional intentions, aligned with aesthetic objectives that might preclude the gamification of participants' interactions. Gamification, however, does not necessarily imply trivialisation and can be represented through the dialectical relationship between paidic (pleasurable, free) and ludic (rule-bound, complex) play. This relationship can be captured and explored within the audience score representation. Similar to the decision-making dial, the balance between paidic and ludic user interfaces can be managed through the networked score implementation.

One possible approach to achieving a more sophisticated UI design in complex scores could involve gradually increasing the complexity of audience involvement during a score performance, allowing ample time for audiences to learn and understand the functionality of the user interface and the interaction model of the score. Similarly, the quantity and intricacy of musical cues linked to graphical elements could be gradually increased once these associations are firmly established through repetition. Conversely, overly simplifying the UI and interaction model may lead to a loss of interest and disengagement amongst audience members. The gradual escalation of ludic play complexity in the score representation should help maintain the required level of audience engagement throughout the performance.

From a compositional perspective, however, all performances worked as intended. The audiences' and musicians' choices were successfully applied in real-time, generating the desired variability in patterns of tension and release. The instrumentation combinations selected by musicians achieved the desired harmonic and textural diversity whilst preserving the intended compositional identity. The audience score view, constructed from the compositional material, effectively conveyed the underlying structure and related ideas, whilst seamlessly blending its audio output with other acoustic and digital sources. The delayed action strategy employed in *Union Rose* was an integral part of the compositional intentions, and the resulting outcomes were fully in line with compositional expectations.

During the *Union Rose* workshops, musicians expressed a desire for a more integrated view of what the audience could see and do. This request was incorporated into *Socket Dialogues*, where the audience voting metre was made visible to all participants, creating a shared dynamic state view. The shared state contributed to the perception of the integrated performance environment.

This visual integration, coupled with the audio generated by the audience, had an implicit influence on the musicians' output, resulting in subtle variations in dynamics and timbre, especially when a negative audience vote led to audible white noise.

After receiving several negative comments regarding the free positioning of mixed notation in *Vexilla*, a decision was made to standardise the placement of various notation elements. For example, dynamics information was consistently placed below pitch notation, whilst notation for timbral techniques was placed above it. Furthermore, variations of graphic clefs for different instruments were replaced with a generic pitch stave layout in *Union Rose* and *Socket Dialogues*. These alterations resulted in predominantly positive feedback from musicians regarding the layout and functionality of the ZScore dynamic notation.

The segmentation of the score layout facilitated the creation of interactive notation overlays, which proved to be an effective means of controlling the improvisation decision-making dial (Figure 1.3). Simple modifications of performance elements in the *Comprov* workshop led to significant variations in sound texture, inciting creative responses from the musicians. The repetition of familiar notation with varying overlay adjustments prompted different interpretations with each pass. ZScore proved to be an effective platform for improvised music-making.

Socket Dialogues also introduced a novel pitch line notation technique, which received mixed feedback from the musicians. Due to its unfamiliarity, sight-reading proved to be more difficult than anticipated. The *Melody* dialogue, in particular, presented a challenge for musicians due to its dense pitch notation. The pitch line notation worked best when combined with indeterminate pitch phrasing notation, outlining only the contour of the pitch movement. Despite the initial apprehension expressed by musicians, *Socket Dialogues* were successfully performed by different instrumentations, albeit with varied accuracy of the notation interpretation. The pitch line notation has the potential to significantly simplify improvisation-oriented scores, but it would require more rehearsal time and effort from the musicians.

The democratisation of decision-making and sound production has inevitably induced the variability of output quality across various performances. The audience's size and level of engagement have affected the overall sound texture in sections where audience mobile devices generated audio output. Nevertheless, compositional strategies and default system behaviours have ensured an uninterrupted performance flow and presentation of compositional intentions, irrespective of the participants' level of engagement.

The accuracy of the score interpretation depended on the level of the musicians' preparedness and the degree of sound production freedom given to musicians. Arguably, the most accurate rendition of the ZScore notation can be heard in the MCME interpretation of *Ukodus*, a fully notated score without any interactive elements. As the portfolio evolved, the level of notational specificity gradually decreased, allowing performers greater interpretive freedom, particularly in *Socket Dialogues*. However, recordings of live performances involving different musicians and audiences demonstrate that the compositional identity was maintained across all performances. Despite differing interpretations, each dialogue retained its intended character through the

precise timing, dynamics, and timbral characteristics of each phrase, supported by consistent digital audio output. In this context, the role of the composer was primarily focused on organising and defining the totality of the output rather than the intricacies of individual sounds, in line with the principles of Gestalt theory.

The process of concept-driven composition has evolved to incorporate elements of front-end design and interactive behaviour modelling. The application of metamodern dialectics, which explores the interplay between subjective and objective methods, has been maintained in all aspects of the integrated composition approach. Starting point concepts, such as flag shapes in *Vexilla*, sudoku patterns in *Ukodus*, window layout in *Union Rose*, or Plato's dialogue structure in *Socket Dialogues*, were mapped to the compositional material and structure through immediate cognitive responses and objective transformational methods.

The role of the composer-performer gradually evolved to encompass active participation in the musical output, particularly in *Comprov* and *Socket Dialogues*. The composer-performer had the ability to shape a performance in real-time by modifying notational overlays, tempo or presets, which had a direct impact on the musicians' and audience's score representations, as well as the audience's audio output. Additionally, the composer-performer had the capability to generate and shape digital audio by manipulating the Max patch directly via MIDI controllers, as demonstrated in the *Socket Dialogues* Interludes.

Both *Union Rose* and *Socket Dialogues* feature a blend of acoustic and digital sound sources, including audio generated by the audience's mobile devices. In live performances, the acoustic instruments were intentionally not amplified to ensure that each sound source's location could be easily identified. The frequency spectrum of the digital audio was adjusted to complement the spectral characteristics of the acoustic sources. Additionally, the combined dynamics of the acoustic instruments and MAX audio had to be balanced to provide sufficient headroom for the mobile device speakers. This approach resulted in an immersive and clear electro-acoustic sound mix, which received mainly positive feedback from the participants.

Chapter 5

Conclusions

The implementation of the ZScore system, together with the accompanying portfolio of works, demonstrates the remarkable potential of computer networking technology to transform processes of music composition and performance. Although initially explored within the context of contemporary experimental music, the system's implementation can be seamlessly adapted to diverse genres and multi-media performance environments.

The research questions, along with the compositional and technical objectives outlined at the outset (Section 1.8), have been addressed and implemented during this research. Traditional music-making relationships have been redefined by expanding the performance participation model to incorporate interactive real-time communications between all participants. The concept of a music score has evolved from a static notation container to a collection of data and algorithms capable of generating dynamic score representations specific for different participant types. ZScore's dynamic notation overlays and interaction strategies successfully enabled improvised music-making with unconstrained positioning of the decision-making dial in real-time.

A different musical aesthetic emerged from technical innovations, where the evaluation of the overall performance output depends on how well the outcomes of actions align with participants' expectations. The role of the composer expanded to encompass a range of technical and performative tasks, whilst the democratisation of decision-making and notation interpretation shifted the composer's focus to the definition of the composition identity and the totality of the output (Gestalt).

The balance between system usability and compositional complexity remains a challenge that needs addressing in each score. The key to successful audience engagement lies in implementing clear conceptual mapping between the graphical score representation and the musical context. Complex action outcomes require a gradual learning curve during a performance, enabling a progressive contextualisation of the participant's role in the interaction model. Furthermore, the time required for participants to switch their focus between musical, visual, and gestural elements needs to be included in the compositional structure. A successful score implementation should effectively convey compositional intentions and create an intuitive and immersive experience for the majority of audience members.

The ZScore system performed remarkably well in all tests and live performances, however, it has only been deployed in small-scale venues so far. Although the system tests indicate that the current implementation should work in a range of different scenarios, the system performance needs to be validated in larger scale deployments. The process of authoring scores for the ZScore system is time-consuming and demands significant technical know-how and effort. To facilitate wider adoption of networked music-making systems, the development of standardised user-friendly composition, performance, and system management tools is necessary.

Participants' feedback from various workshops and live performances indicates evident enthusiasm and willingness from both musicians and audiences to engage and participate in real-time interactive music-making. However, realising this potential relies on the availability of accessible, scalable, and robust networked composition and performance tools. The lessons learned from the development of the ZScore system serve as a stepping stone towards achieving more streamlined and widely adopted networked music-making solutions in the future.

References

- Agostini, A. and D. Ghisi (2012). “Bach: An environment for computer-aided composition in Max”. In: *International Computer Music Conference*, pp. 373–378. URL: <https://www.andreaagostini.eu/wp-content/uploads/2015/08/bach-an-environment-for-computer-aided-composition-in-max.pdf>.
- Apache Software Foundation (1995). *About the Apache HTTP Server Project*. URL: https://httpd.apache.org/ABOUT_APACHE.html (visited on June 5, 2023).
- (1998). *Apache JMeter*. URL: <https://jmeter.apache.org/> (visited on June 5, 2023).
- Apple Inc. (1993). *Logic Pro*. URL: <https://www.apple.com/uk/logic-pro/> (visited on June 5, 2023).
- (2002). *Bonjour zero-configuration networking*. URL: <https://developer.apple.com/bonjour/> (visited on June 5, 2023).
- Avid Technology Inc. (1989). *Pro Tools Music Production*. URL: <https://www.avid.com/pro-tools> (visited on June 5, 2023).
- Azzigotti, L. (2009). *Video: Spam, Generative Score for six performers, thingNY, New York*. URL: <https://youtu.be/9U0Jb-7jRs4> (visited on June 5, 2023).
- Bell, J. (2021). “Distributed Notation in the Browser, an Overview”. In: *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR’20/21*. Hamburg University for Music and Theater, pp. 251–259. ISBN: 978-3-00-066930-9. URL: https://www.tenor-conference.org/proceedings/2021/07_Bell_tenor21.pdf.
- Berners-Lee, T. (1991). *The Original HTTP as defined in 1991*. URL: <https://www.w3.org/Protocols/HTTP/AsImplemented.html> (visited on June 5, 2023).
- Bhagwati, S. (2013a). “Comprovisation – concepts and techniques”. In: *Re) Thinking Improvisation*. URL: https://www.researchgate.net/publication/262490972_Rethinking_Improvisation_Artistic_explorations_and_conceptual_writing.
- (2013b). “Notational perspective and improvisation”. In: *Sound & Score. Essays on Sound, Score and Notation*. Ed. by K. C. Paulo de Assis William Brooks, pp. 165–177. URL: https://www.academia.edu/25207289/Sound_and_Score_Essays_on_Sound_Score_and_Notation.
- Blumröder, C. von (1984). “Offene Form”. In: *Handwörterbuch der musikalischen Terminologie*. Ed. by H. H. Eggebrecht. Steiner, Stuttgart. URL: <https://www.musicconn.de/Vta2/bsb00070512f465t474/hmt:hmt2bsb00070512f465t474?page=465&c=solrSearchHmT>.

- Borio, G. and A. Carone (2018). *Musical improvisation and open forms in the age of Beethoven*. eng. Milton Park, Abingdon, Oxon: Routledge. ISBN: 9781138222960; 1138222968. URL: <https://www.routledge.com/Musical-Improvisation-and-Open-Forms-in-the-Age-of-Beethoven/Borio-Carone/p/book/9780367884628>.
- Bova, T., T. Krivoruchka, and C. Systems (1999). *Reliable UDP protocol*. URL: <https://datatracker.ietf.org/doc/html/draft-ietf-sigtran-reliable-udp-00/> (visited on June 5, 2023).
- Bown, O., B. Carey, and A. Eigenfeldt (Aug. 2015). “Manifesto for a Musebot Ensemble: A platform for live interactive performance between multiple autonomous musical agents”. In: *Proceedings of the ISEA2015: 21st International Symposium on Electronic Art*, pp. 637–644. URL: https://www.isea-archives.org/docs/2015/proceedings/ISEA2015_proceedings.pdf.
- Brown, E. (1954). “*Folio and 4 Systems*” *prefatory note*. URL: <https://earle-brown.org/wp-content/uploads/Folio-and-Four-Systems-Prefatory-Note.pdf> (visited on June 5, 2023).
- (2008). “On December 1952”. In: *American Music* 26.1, pp. 1–12. ISSN: 07344392, 19452349. URL: <http://www.jstor.org/stable/40071686> (visited on June 11, 2023).
- Cassidy, A. (2020). *Video: Second String Quartet, Score Follower*. URL: <https://youtu.be/I500YLLXHvI> (visited on June 5, 2023).
- Cerf, V. and R. Kahn (1974). “A Protocol for Packet Network Intercommunication”. In: *IEEE Transactions on Communications* 22.5, pp. 637–648. DOI: 10.1109/TCOM.1974.1092259.
- Clark, D., K. Pogran, and D. Reed (Dec. 1978). “An introduction to local area networks”. In: *Proceedings of the IEEE* 66, pp. 1497–1517. DOI: 10.1109/PROC.1978.11152.
- Coltrane, J. (1964). *Guide to the John Coltrane Music Manuscript*. URL: <https://sova.si.edu/record/NMAH.AC.0903> (visited on June 5, 2023).
- Constanzo, R. (2014). *Dfscore: networked notation software*. last accessed: 10 Jun 2020. URL: <http://www.rodriigoconstanzo.com/dfscore/>.
- (2019). *Making decisions in time*. last accessed: 10 Jun 2023. URL: <http://www.rodriigoconstanzo.com/2015/04/making-decisions-in-time/>.
- Costa, L. (1998). “Open Systems Interconnect (OSI) Model”. In: *JALA: Journal of the Association for Laboratory Automation* 3.1, pp. 28–35. URL: <https://doi.org/10.1177/221106829800300108>.
- Crnačka Ganga (2014). *Video: “Crnačka Ganga” singing style*. URL: https://youtu.be/VTRb_FMxV28 (visited on June 5, 2023).
- Crockford, D. (2001). *Introducing JSON*. URL: <https://www.json.org/json-en.html> (visited on June 5, 2023).
- Cycling ’74 (2003). *Jitter - Visuals for Max*. URL: <https://cycling74.com/products/jitter> (visited on June 5, 2023).
- Didkovsky, N. and P. Burk (2004). *Java Music Specification Language*. URL: <https://www.algomusic.com/jmsl/jmslhome.html> (visited on June 5, 2023).

- Didkovsky, N. and G. Hajdu (Aug. 2008). “MaxScore: Music Notation in Max/MSP”. In: *Proceedings of the International Computer Music Conference*, pp. 483–486. URL: <http://www.georghajdu.de/gh/fileadmin/material/articles/cr1303.pdf>.
- Dyck, E. V. et al. (2013). “The Impact of the Bass Drum on Human Dance Movement”. In: *Music Perception: An Interdisciplinary Journal* 30.4, pp. 349–359. ISSN: 07307829, 15338312. URL: <http://www.jstor.org/stable/10.1525/mp.2013.30.4.349> (visited on June 24, 2023).
- Eich, B. (1995). *Netscape and Sun Announce JavaScript, the Open, Cross-platform Object Scripting Language For Enterprise Networks and the Internet*. URL: <https://web.archive.org/web/20070916144913/https://wp.netscape.com/newsref/pr/newsrelease67.html> (visited on June 5, 2023).
- Elson, J., L. Girod, and D. Estrin (Dec. 2003). “Fine-grained network time synchronization using reference broadcasts”. In: *ACM SIGOPS Operating Systems Review* 36.SI, pp. 147–163. ISSN: 0163-5980. DOI: 10.1145/844128.844143. URL: <https://doi.org/10.1145/844128.844143>.
- EndRun Technologies (2002). *Precision Time Protocol*. URL: <https://endruntechnologies.com/pdf/PTP-1588.pdf> (visited on June 5, 2023).
- Evans, C., B. Ingerson, and O. Ben-Kiki (2004). *YAML Ain’t Markup Language*. URL: <https://yaml.org/> (visited on June 5, 2023).
- F5 Nginx (2004). *Open Source Nginx*. URL: <https://nginx.org/en/> (visited on June 5, 2023).
- Farrand, P. and J. Borowicz (1988). *Finale music notation software*. URL: <https://www.finalemusic.com/> (visited on June 5, 2023).
- Finn, B. and J. Finn (1993). *Sibelius: Music notation software for everyone*. URL: <https://www.avid.com/sibelius> (visited on June 5, 2023).
- Fober, D., Y. Orlarey, and S. Letz (2013). “Programming Interactive Music Scores with INScore”. In: *Proc. SMC Sound and Music Computing conference*, pp. 185–190. URL: <https://hal.science/hal-00851956>.
- Fober, D. et al. (2010). “Time Synchronization in Graphic Domain - A new paradigm for Augmented Music Scores”. In: *International Computer Music Conference*. Ed. by ICMA. New York, United States, pp. 458–461. URL: <https://hal.archives-ouvertes.fr/hal-02158957>.
- Fober, D. et al. (2021). “A Web Based Environment Embedding Signal Processing in Musical Scores”. In: *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR’20/21*. Hamburg, Germany: Hamburg University for Music and Theater, pp. 221–226. ISBN: 978-3-00-066930-9. URL: https://www.tenor-conference.org/proceedings/2021/03_Fober_tenor21.pdf.
- Forte, A. (1973). *The Structure of Atonal Music*. Yale University Press. ISBN: 9780300021202. URL: <https://yalebooks.yale.edu/book/9780300021202/the-structure-of-atonal-music/>.
- Freed, A. and M. Wright (2002). *Open Sound Control*. URL: <https://opensoundcontrol.stanford.edu/> (visited on June 5, 2023).
- Freed, A. et al. (2014). “o.io: a Unified Communications Framework for Music, Intermedia and Cloud Interaction”. In: *Music Technology meets Philosophy - From Digital Echos to*

- Virtual Ethos: Joint Proceedings of the 40th International Computer Music Conference, ICMC 2014, and the 11th Sound and Music Computing Conference, SMC 2014, Athens, Greece, September 14-20, 2014.* Michigan Publishing. URL: <https://hdl.handle.net/2027/spo.bbp2372.2014.241>.
- FromTextToSpeech.com (2020). *Free online Text To Speech (TTS) service with natural sounding voices.* URL: <http://fromtexttospeech.com/> (visited on June 5, 2023).
- Fry, B. and C. Rea (2001). *Welcome to Processing!* URL: <https://processing.org/> (visited on June 5, 2023).
- Good, M. (2001). *MusicXML.* URL: <https://www.musicxml.com/> (visited on June 5, 2023).
- Gosling, J. and H. McGilton (1996). *The Java Language Environment: Contents.* URL: <https://www.oracle.com/java/technologies/language-environment.html> (visited on June 5, 2023).
- Gottfried, R. (2015). “SVG to OSC Transcoding as a Platform for Notational Praxis and Electronic Performance”. In: *Proceedings of the First International Conference on Technologies for Music Notation and Representation – TENOR’15.* Paris, France, pp. 154–161. ISBN: 978-2-9552905-0-7. URL: <https://www.tenor-conference.org/proceedings/2015/25-Gottfried-SVG-OSC-Notation.pdf>.
- (2022). “Symbolist Re-Imagined: Bidirectional Graphic-Semantic Mapping For Media Notation Authoring And Performance”. In: *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR’2022.* Marseille, France: PRISM Laboratory, pp. 92–101. ISBN: 979-10-97498-03-0. URL: https://www.tenor-conference.org/proceedings/2022/TENOR_2022_paper_23.pdf.
- Gottfried, R. and G. Hajdu (2019). “Drawsocket: A Browser Based System for Networked Score Display”. In: *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR’19.* Melbourne, Australia: Monash University, pp. 15–25. ISBN: 978-0-6481592-5-4. URL: <https://www.tenor-conference.org/proceedings/2019/05Gottfried.pdf>.
- Grame CNCM (2002). *Faust: Functional Programming Language for Real Time Signal Processing.* URL: <https://faust.grame.fr/> (visited on June 5, 2023).
- GreenSock Inc. (2014). *GSAP: Professional-grade JavaScript animation for the modern web.* URL: <https://greensock.com/gsap/> (visited on June 5, 2023).
- Guido Van Der Werve (2007). *Video excerpt from “Nummer acht, everything is going to be alright”, published by Francoise Deprez.* URL: <https://youtu.be/5366DD9JauU> (visited on June 5, 2023).
- Hajdu, G. (2005). “Quintet.net: An Environment for Composing and Performing Music on the Internet”. In: *Leonardo Music Journal* 38, pp. 23–30. URL: https://www.researchgate.net/publication/242384302_Quintetnet_An_Environment_for_Composing_and_Performing_Music_on_the_Internet.

- Hayles, K. (1999). *How We Became Posthuman: Virtual Bodies in Cybernetics, Literature, and Informatics*. USA: University of Chicago Press. ISBN: 9780226321455. URL: <https://press.uchicago.edu/ucp/books/book/chicago/H/bo3769963.html>.
- Hedges, S. A. (1978). “Dice Music in the Eighteenth Century”. In: *Music & Letters* 59.2, pp. 180–187. ISSN: 00274224, 14774631. URL: <http://www.jstor.org/stable/734136>.
- Hoadley, R. (2011). *Calder’s Violin*. URL: <https://rhoadley.net/comp/calder/> (visited on June 5, 2023).
- Hochschule für Musik und Theater Hamburg (2019). *Video: Multimedial Concert at Hamburg’s St. Pauli Elbtunnel with 144 musicians - Day 1 Concert 2*. URL: https://youtu.be/9dx7VRAeX_A (visited on June 5, 2023).
- Hoos, H. H. and K. A. Hamel (1997). *The GUIDO Music Notation Format*. URL: <https://guidolib.sourceforge.io/doc/GUIDO-Music%20Notation%20Format.html> (visited on June 5, 2023).
- Hope, C. et al. (2015). “The Decibel ScorePlayer - a digital tool for reading graphic notation”. In: *Int. Conf. on New Tools for Music Notation and Representation - TENOR 2015*, pp. 59–69. URL: <https://www.tenor-conference.org/proceedings/2015/08-Hope-Decibel.pdf>.
- Hope, C. (2011). *Video: Longing, score follow, Candied Limbs*. URL: <https://youtu.be/NXTUn6oAIVo> (visited on June 5, 2023).
- (2019a). *Speechless*. URL: <https://www.cathope.com/art-work-speechless> (visited on June 5, 2023).
- (2019b). *Video: Speechless Score Follow*. URL: <https://vimeo.com/383439576> (visited on June 5, 2023).
- iMatix (2007). *ZeroMQ: An open-source universal messaging library*. URL: <https://zeromq.org/> (visited on June 5, 2023).
- (2021). *100GbE Tests with ZeroMQ v4.3.2*. URL: <http://wiki.zeromq.org/results:100gbe-tests-v432> (visited on June 5, 2023).
- IRCAM (1998). *OpenMusic*. URL: <https://forum.ircam.fr/projects/detail/openmusic/> (visited on Jan. 4, 2024).
- (2018). *@ircam/sync*. URL: <https://www.npmjs.com/package/@ircam/sync> (visited on Jan. 4, 2024).
- James, S. et al. (2017). “Establishing connectivity between the existing networked music notation packages Quintet.net, Decibel Score Player and MaxScore.” In: *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR’17*. Ed. by H. L. Palma et al. Universidade da Coruña, pp. 171–183. ISBN: 978-84-9749-666-7. URL: https://www.tenor-conference.org/proceedings/2017/22_James_tenor2017.pdf.
- Johnson, J. (2015). *Out of Time: Music and the Making of Modernity*. Oxford University Press. ISBN: 9780190233273. URL: <https://books.google.co.uk/books?id=S0ZnBgAAQBAJ>.
- Kastrup, D. et al. (1997). *LilyPond, music notation for everyone*. URL: <https://lilypond.org/> (visited on June 5, 2023).

- Katan, S. (2016). *Conditional Love*. URL: <https://simonkatan.co.uk/projects/conditionallove.html> (visited on June 5, 2023).
- Kennedy, J. (July 2010). “Plato’s Forms, Pythagorean Mathematics, and Stichometry”. In: *Apeiron* 1. DOI: 10.1515/APEIRON.2010.43.1.1.
- Kirby, A. (2015). “The Death of Postmodernism and Beyond”. en. In: 1st ed. *Supplanting the Postmodern : An Anthology of Writings on the Arts and Culture of the Early 21st Century*. New York: Bloomsbury Academic, pp. 49–60. ISBN: 978-1-5013-0690-7. URL: https://philosophynow.org/issues/58/The_Death_of_Postmodernism_And_Beyond.
- Krasner, G. and S. Pope (Jan. 1988). “A cookbook for using the model-view controller user interface paradigm in Smalltalk-80”. In: *Journal of Object-oriented Programming - JOOP* 1. URL: <https://www.lri.fr/~mbl/ENS/FONDIHM/2013/papers/Krasner-JOOP88.pdf>.
- Kraus, N. and J. Slater (2016). “Beyond Words: How Humans Communicate Through Sound”. In: *Annual Review of Psychology* 67.1. PMID: 26361049, pp. 83–103. URL: <https://doi.org/10.1146/annurev-psych-122414-033318>.
- Lévesque, M. and D. Tipper (2016). “A Survey of Clock Synchronization Over Packet-Switched Networks”. In: *IEEE Communications Surveys & Tutorials* 18.4, pp. 2926–2947. DOI: 10.1109/COMST.2016.2590438.
- Lind, A. (2020a). *Mobile phone instruments and scores*. URL: <https://mobilephoneorchestra.com/play-orchestra> (visited on June 5, 2023).
- (2020b). *Towards a Novel Mobile Phone Orchestra Platform*. URL: <https://www.researchcatalogue.net/view/590306/590307> (visited on June 5, 2023).
- Louzeiro, P. (2018). “Improving Sight-Reading Skills through Dynamic Notation - the Case of Comprovisador”. In: *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR’18*. Concordia University, pp. 55–61. ISBN: 978-1-5251-0551-7. URL: https://www.tenor-conference.org/proceedings/2018/08_Louzeiro_tenor18.pdf.
- Lüneburg, B. (2017). “Between “Ludic Play” and “Performative Involvement””. In: *eContact! The electronic journal of electroacoustics* 20.2. URL: https://econtact.ca/20_2/lueneburg_gamifiedmultimedia.html.
- (Dec. 2018). “Between Art and Game: Performance Practice in the Gamified Audiovisual Artworks of GAPP”. In: *The Computer Games Journal* 7.4, pp. 243–260. URL: <https://doi.org/10.1007/s40869-018-0066-7>.
- MacCallum, J. et al. (2015). “Dynamic Message-Oriented Middleware with Open Sound Control and Odot”. In: *International Computer Music Conference*, pp. 58–65. URL: <https://quod.lib.umich.edu/i/icmc/bbp2372.2015.010/1>.
- Malouin, C. (2020). *What is MIDI clock?* URL: <https://support.redpandalab.com/support/solutions/articles/43000586933-what-is-midi-clock-> (visited on Apr. 1, 2024).

- McCarthy, J. (1960). *Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I*. URL: <https://www-formal.stanford.edu/jmc/recursive/recursive.html> (visited on June 5, 2023).
- McCartney, J. (1996). *Supercollider: A Platform For Audio Synthesis And Algorithmic Composition*. URL: <https://supercollider.github.io/> (visited on June 5, 2023).
- McKay, E. N. and D. Musgrave (2018). *Intuitive Design: Eight Steps to an Intuitive UI*. Black Watch Publishing. ISBN: 9780999612507. URL: <https://books.google.co.uk/books?id=QcsetAEACAAJ>.
- Microsoft (1995). *IIS Web Server Overview*. URL: <https://learn.microsoft.com/en-us/iis/get-started/introduction-to-iis/iis-web-server-overview> (visited on June 5, 2023).
- Mills, D. L. (1985). *Network Time Protocol (NTP)*. URL: <https://datatracker.ietf.org/doc/html/rfc958> (visited on June 5, 2023).
- Narveson, J. and D. Trueman (2013). “Landini: a networking utility for wireless lan-based laptop ensembles”. In: *SMC Sound and Music Computing conference*, pp. 309–316. URL: https://jaschanarveson.com/code/landini/LANdini_paper.pdf.
- Nijs, L., M. Lesaffre, and M. Leman (Jan. 2013). “The musical instrument as a natural extension of the musician”. In: *La musique et ses instruments - Music and its instruments*. Delatour France, pp. 467–484. ISBN: 9782752101594. URL: https://www.researchgate.net/publication/257939901_The_musical_instrument_as_a_natural_extension_of_the_musician.
- Nord, H. and E. Chambe-Eng (1995). *Qt Framework*. URL: <https://www.qt.io/product/framework> (visited on June 5, 2023).
- Nyman, M. (1999). *Experimental Music: Cage and Beyond*. Music in the 20th century. Cambridge University Press. ISBN: 9780521653831. URL: <https://books.google.co.uk/books?id=QEBzEhzAkYwC>.
- Oda, R. and R. Fiebrink (2016). “The Global Metronome: Absolute Tempo Sync For Networked Musical Performance”. In: *Proceedings of the International Conference on New Interfaces for Musical Expression*. Brisbane, Australia: Queensland Conservatorium Griffith University, pp. 26–31. ISBN: 978-1-925455-13-7. DOI: 10.5281/zenodo.1176096. URL: http://www.nime.org/proceedings/2016/nime2016_paper0006.pdf.
- OpenJFX (2008). *JavaFx*. URL: <https://openjfx.io/> (visited on June 5, 2023).
- OpenJS Foundation (2009). *About Node.js*. URL: <https://nodejs.org/en/about> (visited on June 5, 2023).
- Opstad, J. A. (2022). “A new kind of score: rethinking the relationship between composer, performer and technology”. PhD thesis. University of Birmingham. URL: <https://etheses.bham.ac.uk/id/eprint/13143/>.
- Oracle Corporation and OpenJDK Community (2012). *Nashorn Engine*. URL: <https://github.com/openjdk/nashorn> (visited on June 5, 2023).
- Pasquier, P. et al. (Jan. 2017). “An Introduction to Musical Metacreation”. In: *Computers in Entertainment* 14.2. DOI: 10.1145/2930672. URL: <https://doi.org/10.1145/2930672>.

- Petrović, A. (2018). *Ankica Petrović The Art of Ganga Singing : Cultural Tradition of the Dinaric Area*. Zagreb: Synopsis. ISBN: 9789537968540. URL: <https://www.synopsisbook.com/knjige/umjetnost-pjevanja-gange-kulturna-tradicija-dinarske-zone/>.
- Pinchot, J., K. Pullet, and D. Rota (Apr. 2011). “How Mobile Technology is Changing Our Culture”. In: *Journal of Information Systems Applied Research* 4, pp. 39–48. URL: https://www.researchgate.net/publication/278005421_How_Mobile_Technology_is_Changing_Our_Culture.
- Pirogov, A. (2017). *Aeron – low latency transport protocol*. URL: <https://medium.com/@pirogov.alexey/aeron-low-latency-transport-protocol-9493f8d504e8> (visited on June 5, 2023).
- Plato, Jowett edition (1892). *The Dialogues of Plato, in 5 vols*. URL: <https://oll.libertyfund.org/title/plato-the-dialogues-of-plato-in-5-vols-jowett-ed> (visited on June 5, 2023).
- Pritchett, J. (1996). *The Music of John Cage*. Music in the 20th century. Cambridge University Press. ISBN: 9780521565448. URL: <https://books.google.co.uk/books?id=riHo22Hi8QAC>.
- Puckette, M. S. (1996). *Pure Data*. URL: <https://puredata.info/> (visited on June 5, 2023).
- Puckette, M. S. and D. Zicarelli (1990). *What is Max?* URL: <https://cycling74.com/products/max> (visited on June 5, 2023).
- Rae, C. (1999). *The Music of Lutosławski*. Omnibus Press. ISBN: 9780711969100. URL: <https://books.google.co.uk/books?id=hYrgaF-0SuEC>.
- Red Hat JBoss (2014). *Undertow web server*. URL: <https://undertow.io/> (visited on June 5, 2023).
- Reed, D. P. and J. Postel (1980). *User Datagram Protocol*. RFC 768. DOI: 10.17487/RFC0768. URL: <https://datatracker.ietf.org/doc/html/rfc768>.
- Schoenberg, A. (1912). *Pierrot Lunaire [w/ full score], video published by incipitsify*. URL: <https://youtu.be/vQVkbKULKpI> (visited on June 5, 2023).
- Small, C. (1998). *Musicking: The Meanings of Performing and Listening*. Music / Culture. University Press of New England. ISBN: 9780819522566. URL: <https://books.google.co.uk/books?id=fYmkAQAACAAJ>.
- Smith, D. J. (1983). *Official MIDI Specifications*. URL: <https://midi.org/specs> (visited on June 5, 2023).
- Spikerog SAS (2018). *Text2speech*. URL: <http://text2speech.org> (visited on June 5, 2023).
- Stockhausen, K. (1963). *Texte zur elektronischen und instrumentalen Musik, Band 1*. DuMont Dokumente: Musik. M.D. Schauberg, pp. 189–210. URL: <https://books.google.co.uk/books?id=aOk5AQAAlAAJ>.
- Tanaka, A. (2019). “Embodied Musical Interaction”. In: *New Directions in Music and Human-Computer Interaction*. Ed. by S. Holland et al. Cham: Springer International Publishing, pp. 135–154. ISBN: 978-3-319-92069-6. URL: https://doi.org/10.1007/978-3-319-92069-6_9.
- Tatar, K. and P. Pasquier (Sept. 2018). “Musical agents: A typology and state of the art towards Musical Metacreation”. In: *Journal of New Music Research* 47, pp. 1–50. doi: 10.1080/09298215.2018.1511736.

- The Adobe Inc. (1987). *Adobe Illustrator*. URL: <https://www.adobe.com/uk/products/illustrator.html> (visited on June 5, 2023).
- The Object Management Group (1997). *Unified Modeling Language (UML)*. URL: <https://www.omg.org/spec/UML> (visited on June 5, 2023).
- The Schoyen Collection (2023). *Oldest Known Music Notation*. URL: <https://www.schoyencollection.com/music-notation/old-babylonia-cuneiform-notation/oldest-known-music-notation-ms-5105> (visited on June 5, 2023).
- Thompson, M. and T. Montgomery (2014). *Aeron: the global technology standard for high-throughput, low-latency, fault-tolerant trading systems*. URL: <https://aeron.io/> (visited on June 5, 2023).
- Thompson, M. et al. (2011). *LMAX Disruptor: High performance alternative to bounded queues for exchanging data between concurrent threads*. URL: <https://lmax-exchange.github.io/disruptor/disruptor.html> (visited on June 5, 2023).
- Uberti, J. and P. Thatcher (2011). *WebRTC: Real-Time Communication in Browsers*. URL: <https://www.w3.org/TR/webrtc/> (visited on June 5, 2023).
- Utz, C. (2017). “Time-Space Experience in Works for Solo Cello by Lachenmann, Xenakis and Ferneyhough: a Performance-Sensitive Approach to Morphosyntactic Musical Analysis”. In: *Music Analysis* 36.2, pp. 216–256. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/musa.12076>.
- Vear, C. (2019). *The Digital Score: Musicianship, Creativity and Innovation*. Routledge. ISBN: 9780429504495. URL: <https://books.google.co.uk/books?id=oSblwQEACAAJ>.
- Vermeulen, T. and R. van den Akker (2010). “Notes on metamodernism”. In: *Journal of Aesthetics & Culture* 2.1, p. 5677. DOI: 10.3402/jac.v2i0.5677. URL: <https://doi.org/10.3402/jac.v2i0.5677>.
- W3C (2001). *Scalable Vector Graphics (SVG)*. URL: <https://www.w3.org/Graphics/SVG/> (visited on June 5, 2023).
- (2008). *HTML 5*. URL: <https://www.w3.org/TR/2008/WD-html5-20080122/> (visited on June 5, 2023).
- (2009). *The WebSocket API*. URL: <https://www.w3.org/TR/2021/NOTE-websockets-20210128/Overview.html> (visited on June 5, 2023).
- (2011). *Web Audio API*. URL: <https://www.w3.org/TR/webaudio/> (visited on June 5, 2023).
- (2012). *Web Speech API*. URL: <https://wicg.github.io/speech-api/> (visited on June 5, 2023).
- (2017). *WebAssembly*. URL: <https://webassembly.org/> (visited on June 5, 2023).
- Wang, G. (2003). *ChuckK: Music Programming Language*. URL: <https://chuck.stanford.edu/> (visited on June 5, 2023).
- Wang, G., G. Essl, and H. Penttinen (Mar. 2014). “The Mobile Phone Orchestra”. In: *The Oxford Handbook of Mobile Music Studies, Volume 2*. Oxford University Press. ISBN: 9780199913657. URL: <https://doi.org/10.1093/oxfordhb/9780199913657.013.018>.

- Web Hypertext Application Technology Working Group (2006). *Server-sent events specification*.
 URL: <https://html.spec.whatwg.org/multipage/server-sent-events.html> (visited on June 5, 2023).
- Weinberger, N. M. (2004). “Music and the Brain”. In: *Scientific American* 291.5, pp. 88–95. ISSN: 00368733, 19467087. URL: <http://www.jstor.org/stable/26060767> (visited on June 11, 2023).
- Winett, J. M. (1971). *The Definition of a Socket*. URL: <https://www.rfc-editor.org/rfc/rfc147> (visited on June 5, 2023).
- Wölfflin, H. et al. (1915). *Principles of Art History: The Problem of the Development of Style in Early Modern Art*. Texts & Documents. Getty Research Institute. ISBN: 9781606064528. URL: <https://books.google.co.uk/books?id=IGr1CAAQBAJ>.
- Wood, R. W. (1946). “Concerning “Sprechgesang””. In: *Tempo* 2, pp. 3–6. ISSN: 00402982, 14782286. URL: <http://www.jstor.org/stable/943969>.
- Zagorac, S. (2020). *Video: ZScore Dynamic Notation*. URL: <https://youtu.be/Yh6wUqLZwkU> (visited on June 5, 2023).
- Zagorac, S. and P. Alessandrini (2018). “ZScore: A Distributed System For Integrated Mixed Music Composition and Performance”. In: *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR’18*. Montreal, Canada: Concordia University, pp. 62–70. ISBN: 978-1-5251-0551-7. URL: https://www.tenor-conference.org/proceedings/2018/09_Zagorac_tenor18.pdf.
- Zagorac, S. and M. Zbyszynski (2020). “Networked Comprovisation Strategies with ZScore”. In: *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR’20/21*. Hamburg, Germany: Hamburg University for Music and Theater, pp. 133–140. ISBN: 978-3-00-066930-9. URL: https://www.tenor-conference.org/proceedings/2020/18_Zagorac_tenor20.pdf.

Appendix A

ZScore Server Testing

A.1 Testing Scope

The purpose of this testing was to verify ZScore's web server behaviour under typical use case scenarios and find system performance limits by increasing the web traffic load until the system's breaking point is found (i.e. system load and stress tests). ZScore's web front end is currently used to render score views on the audience's mobile devices. Under typical load conditions, the system is expected to serve up to 100 connected web interface users without any network packet loss and any adverse impact on the internal network score distribution. The maximum allowed web page load time in these tests was set to 5 seconds, whilst the maximum server state data push latency was set to 1 second. The tests included client connectivity over wired (Ethernet) and wireless (Wi-Fi) connections for both HTTP and WebSocket protocols. Furthermore, a set of tests was devised to assess the impact of firewall filtering on the system throughput. The testing scope did not include wireless access point range tests and Open Sound Control UDP messaging to musicians' front ends.

A.2 Testing Environment

The ZScore web server is currently integrated into the ZScore application written in Java (version 8). The server utilises the Undertow library (v2.0.1) to provide HTTP, WebSockets, and SSE service. For this test, the ZScore application was run on a 16" MacBook Pro laptop connected to the Mikrotik hAP ac router via Ethernet cable, as illustrated in Figure A.1. The test clients (custom Java and Apache JMeter) were executed on a separate MacBook Pro laptop (Figure A.1). The testing client laptop either connected via wired Ethernet or wirelessly to the router, depending on the specific test scenario.

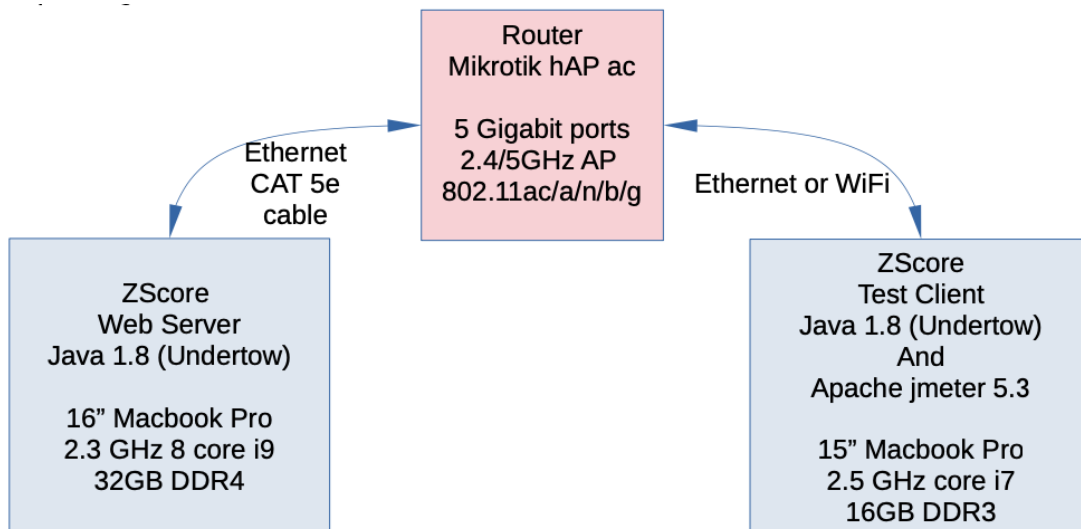


Figure A.1: Test Environment

A.2.1 Router Setup

In a public performance situation, allowing the audience’s mobile device access to ZScore’s internal network presents a potential security risk. To address this, a network firewall, an industry-standard protection mechanism, was employed to isolate the internal network and limit external users’ access. To assess the impact of the router’s firewall on both wired and wireless connections, a separate network bridge (guestBridge) with its own IP address subnet range was created on the router. One of the Ethernet ports on the router (port 5) and both virtual wireless Guest WLANs (wlanGuest and wlanGuest5g) were assigned to the created guest bridge (Figure A.2).

A set of firewall rules were then implemented to prevent guest bridge access to all standard ports on the internal network, except for DNS (port 53) and HTTP (port 80). Additionally, a network address translation (NAT) rule was added to route all HTTP requests originating from the guest bridge subnet to the ZScore server, as depicted in Figures A.3 and A.4.

In order to limit the bandwidth available to guest users and prevent network congestion, two router queues were created – one for internal connections and another for guest connections (Figure A.5). Both queues were configured to use the SFQ (Stochastic Fairness Queuing) algorithm, ensuring a fair round-robin distribution to all sub-streams. After conducting initial testing and calculations, I determined that limiting guest users to 200 Mbps overall bandwidth would be sufficient for the typical use case scenario.

A.2.2 Wi-Fi Setup

The Mikrotik hAP ac router provides two modes: 2.4 GHz (b/g/n) and 5 GHz (a/n). For this test, both WLANs (2.4 and 5 GHz wireless LANs) were configured with the same SSID, allowing the test client’s wireless adapter to choose the best connection mode automatically. In a real-life

	#	Interface	Bridge	Horiz...	Trust...	Priority (hex)	Path Cost	
;;; defconf								
- D	H	0	ether2	bridge		no	80	10
;;; defconf								
- D	IH	1	ether3	bridge		no	80	10
;;; defconf								
- D	IH	2	ether4	bridge		no	80	10
;;; Guest port5 VLAN2								
- D	I	3	ether5	bridgeGuest		no	80	10
;;; defconf								
- D	I	4	sfp1	bridge		no	80	10
;;; defconf								
- D	I	5	wlan1	bridge		no	80	10
;;; defconf								
- D		6	wlan2	bridge		no	80	10
;;; Guest Wifi Port								
- D	I	7	wlanGuest	bridgeGuest		no	80	10
;;; Guest Wifi Port								
- D	I	8	wlanGuest5g	bridgeGuest		no	80	10

Figure A.2: Router bridge ports assignment

;;; Guest Allow DNS										
- D		13	✓ accept	forward			17 (udp)		53	bridgeGuest
- D		14	✓ accept	forward			6 (tcp)		53	bridgeGuest
- D		15	✓ accept	forward			17 (udp)		53	bridgeGuest
- D		16	✓ accept	forward			6 (tcp)		53	bridgeGuest
;;; Guests block local ports										
- D		17	✗ drop	input		10.2.2.1	6 (tcp)		20,21,22,23,25,8291	

Figure A.3: Guest bridge firewall rules

;;; Guest Redirect DNS										
- D		4	⇄ redirect	dstnat		10.2.2.0/24		17 (udp)	53	
- D		5	⇄ redirect	dstnat		10.2.2.0/24		6 (tcp)	53	
;;; Guest bridge nat to web server										
- D		6	⇄ dst-nat	dstnat		10.2.2.0/24		6 (tcp)	80	bridgeGuest

Figure A.4: Guest bridge NAT rules

	#	Name	Target	Upload Max Limit	Download Max Limit
- D	0	GuestQueue	10.2.2.0/24	150M	150M
- D	1	MainQueue	192.168.88.0/24	250M	250M

Figure A.5: Router Queue setup

performance scenario, audience mobile devices will be allowed to select the wireless connection mode themselves, simplifying the user connection process.

The router's channel selection setting for both 2.4 and 5 GHz WLANs was set to 'auto',

allowing the router's algorithm to automatically select the least congested channel. The 2.4GHz WLAN channel width was set to 20 MHz, whilst the 5 GHz WLAN channel width was set to 20/40 MHz XX, where the XX mode allows for automated control channel frequency selection. According to the Mikrotik product specifications, the maximum theoretical hAP ac throughput for the 2.4 GHz mode is 450 Mbit/s, and for the 5 GHz mode, it is 1300 Mbit/s. The actual maximum throughput achieved during testing was around 120 Mb/s, which aligns with expectations as the maximum theoretical throughput for a single wireless stream is 150 Mbps. The Mikrotik AP provides multiple streams (MIMO), so when multiple Wi-Fi clients are connected, the overall throughput should increase by up to five times (2x at 2.4 GHz and 3x at 5 GHz).

Significant congestion and interference from neighbouring wireless access points were encountered during testing. A total of 34 access points with better than -75 dBm wireless signal strength were scanned from the testing site. Given the significant interference and congestion experienced from neighbouring wireless access points during testing, it is unlikely that actual performance sites would face worse interference issues.

A.3 Testing Procedure

Several testing scenarios were designed to cover all permutations within the testing scope. These permutations involved both wired and wireless test client connections on both the guest and internal bridge. Each scenario was executed five times to obtain average results and ensure statistical reliability.

The *Union Rose* score was loaded into the server as a typical example of the scores written for the ZScore system. The initial HTTP load for the audience score was approximately 1.9 MB of data, which included required HTML, Javascript, CSS, and MP3 files. The average server state data update size was approximately 18KB.

The firewall contained 20 filter rules, set up as described above. During tests where the client was connected through the guest bridge, all data had to pass through the firewall. However, if the test client was connected through the internal bridge, the data bypassed the firewall. Additionally, wired connections on the internal bridge benefitted from the hardware switch optimisation.

For the wireless tests, the client host was positioned approximately 2 metres away from the access point. As described above, the scope did not include wireless range testing.

Two types of testing clients were used: custom Java (Undertow) and Apache JMeter (1998). These clients were chosen to simulate different types of user interactions and load scenarios on the ZScore web server.

A.3.1 JMeter Tests

ZScore's web client uses the HTTP polling mechanism when neither SSE nor WebSocket transports are available on the client-side. The JMeter tests cover this worst case scenario. The

client's behaviour was first recorded through the JMeter proxy, starting with the retrieval of the index page, which included HTML, Javascript, CSS, and MP3 content (1.9 MB overall). The captured scenario then involved issuing 23 HTTP polling requests for the server state data at 500 millisecond intervals. Each server data state update size was approximately 18 KB. The ramp-up time to reach the desired client number was set to 10 sec. After the ramp-up time, all clients simultaneously issued the recorded requests. The JMeter tests were primarily used to stress test the ZScore server and determine its system limits.

A.3.2 Custom Undertow Client Tests

The custom Java tests encompassed both typical HTTP client content retrieval and WebSocket server data push scenarios.

The HTTP tests first created the specified number of HTTP clients, and then index HTML page requests were issued for each client to download the initial content (HTML, Javascript, CSS, and MP3 files, with an overall size of ~1.9 MB). The index page requests were staggered to mimic real-life scenarios. A random number of clients (between 1 and 5) simultaneously requested the index page. After a random time interval (between 0 and 10 seconds), the next group of clients was randomly selected to send their requests. This process was repeated until all test clients had submitted index page requests. For 200 HTTP clients, the average duration of the test was around 50 seconds.

Similarly, for the WebSocket tests, the required number of clients were initialised, and they listened for server state updates. In these tests, the ZScore server replayed server-side events for the first 16 beats of *Union Rose*. This involved 8 server state data updates, each approximately 18 KB in size. The duration of a single test was approximately 12 seconds.

A.4 Findings And Analysis

Generally, the test results confirmed expected performance patterns. The most important findings can be outlined as follows:

- In its current form, the **ZScore server** can handle a significantly higher load than the typical use case scenario. Test results indicate that network data back pressure and packet loss begin to occur at around **1500 concurrent users**.
- The **router's switch** can successfully serve **200 concurrent wired connections** with no firewall (WebSocket: average 8 ms, 90th percentile 19 ms, max 253 ms) and going through the firewall, without any packet loss (WebSocket: average 40 ms, 90th percentile 114 ms, max 869 ms).
- The **router's wireless AP** can successfully serve **up to 100 wireless users** connected through the firewall. The maximum number of Wi-Fi connections that did not suffer any

packet loss was 125. However, the max latency for 100 wireless connections reached 9 sec (average: 1.6 sec, 90th percentile: 3.3 sec).

- Firewall filtering had a notable impact on wired clients' throughput. However, for the wireless connections, the benefits of a secure firewall outweigh a negligible performance difference between open and firewalled connections.
- **Wireless** tests latency **variance** was almost **4 times higher** than the wired tests variance. The most likely cause for such a large difference was the significant interference from the neighbouring wireless access points during testing, as described above.

The most significant difference between a real-life performance situation and these tests is that, in the real world, there would be multiple hosts (mobile devices) making a single connection to the server. This difference could have multiple consequences. The AP would have to manage and route a number of connections which might impede its performance. On the other hand, the router's multiple wireless streams (2 for 2.4 GHz and 3 for 5 GHz) would be better utilised when multiple client adapters are connected. This could theoretically increase available bandwidth fivefold. The only way to measure this impact would be to have multiple (50+) mobile devices connected to the router's AP in a controlled environment.

In wired HTTP tests, it was not possible to connect more than 235 clients, and in wireless tests, the maximum was 150 clients. The initial suspicion was that this was due to the test laptop's operating system file descriptor limits. However, even after increasing the number of OSX file descriptors, the tests failed at the same point. As JMeter tests managed successfully to create 2000 connections, it is likely that this is a limitation of the Undertow client implementation (multiple non-blocking threads and OS resource utilisation).

The ZScore's server memory heap size never went over 250 MB during the testing, indicating a relatively light memory usage and no significant memory leaks whilst handling the web traffic. The maximum JVM CPU load during testing was 1.5% which, again, indicates a fairly low server-side load.

The Mikrotik Queue size reached its peak at around 110 Mbps for 100 connected users. This aligns with expectations as each connection had to download 1.9 MB of initial files, and the limitation of a single wireless stream is 150 Mbps. It is essential to monitor the queue size whenever the maximum throughput or the number of connected users changes significantly.

In a real-life situation, a Wi-Fi user's experience will depend heavily on the performance site's wireless channel congestion. However, it is unlikely that the throughput will be much worse than the testing data numbers due to the reasons described above.

A.5 Scalability Ramifications And Possible Solutions

From the ZScore web user's perspective, two factors can impact the system's performance the most: the size of the data to push through the network, and the number of connected users. If

it is assumed that *Union Rose* is a typical score and represents the data load used in the ZScore system, then the number of connected users becomes critical for system scalability planning. The system design discussion could be framed around the number of expected users (audience members) as follows.

A.5.1 System Design For Up To 100 Wi-Fi Users

Based on the test results, current system architecture should handle 100 users connected over Wi-Fi and going through the firewall with good performance.

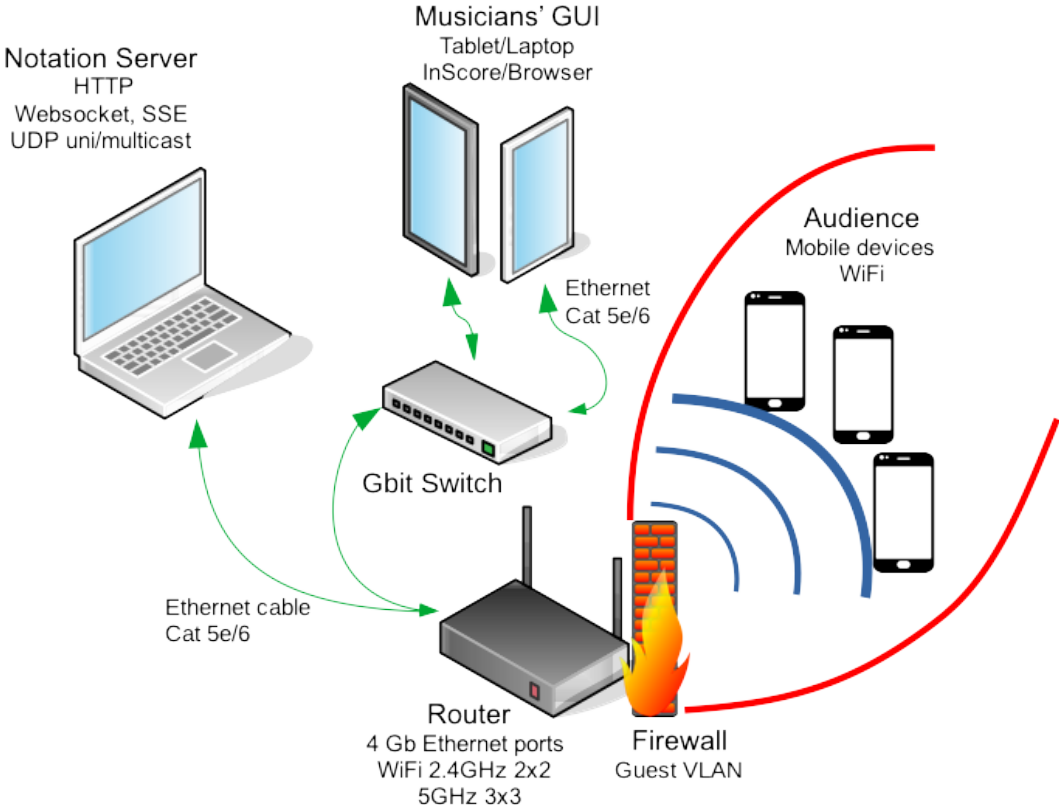


Figure A.6: System architecture for up to 100 Wi-Fi users

A.5.2 System Design For 200 To ~500 Wi-Fi Users

For 200 connected Wi-Fi users, the typical required throughput would increase to around 400 Mbps. Although Mikrotik's theoretical wireless throughput is much higher, the tests have shown that additional wireless access points might be required to support this load effectively. Based on the testing results, it could be extrapolated that for each additional 100 users, one more wireless AP of similar specifications to the test AP is required. Alternatively, a more powerful MU MIMO AP (such as Unifi UAP AC HD) could be used to cater for more than 100 users. This expansion could be done until the router's switch becomes a bottleneck. By extrapolating

test results, it can be concluded that the current router could serve up to 500 users, which would require 4 additional MIMO or 2-3 MU MIMO access points. A system design suggestion for up to 200 connected Wi-Fi users is shown in Figure A.7.

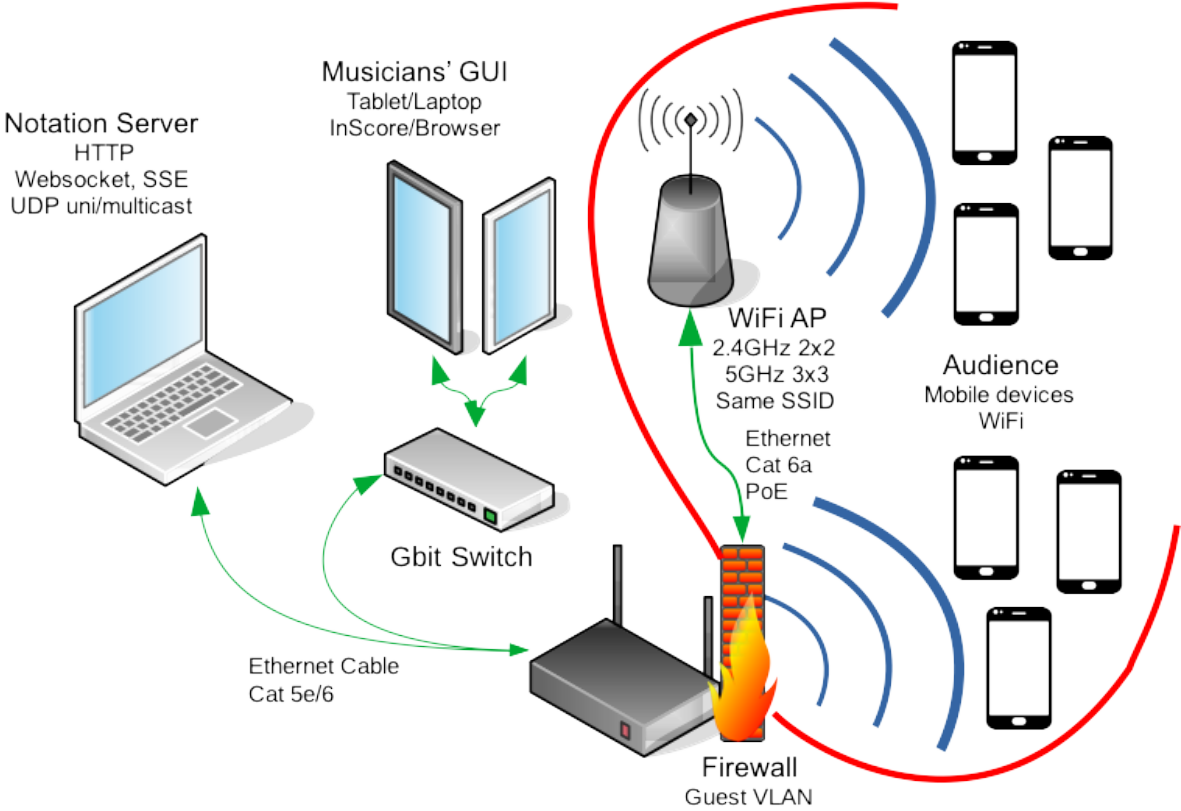


Figure A.7: System design for up to 200 Wi-Fi users

Any additional AP needs to be connected to the router via an Ethernet cable for stability and sufficient throughput. A shielded CAT 6e Ethernet cable would be a good choice for this purpose, as it provides reduced interference and high throughput (up to 10 Gb). Furthermore, CAT 6a can feed power over Ethernet (PoE) to AP if required. PoE would simplify system installation, as it replaces conventional power cabling and allows for easier deployment of the access points without the need for separate power outlets.

Additional APs should have similar specifications to hAP ac AP (MIMO 2x2 at 2.4 GHz and 3x3 at 5 Ghz) or better in order to provide required throughput (e.g. Unifi AP AC PRO). The additional AP should be set up with the same guest SSID as the router's AP to simplify the user connection procedure. In this design, there are no guarantees as to which access point mobile devices will connect to. However, most modern Wi-Fi adapters have built-in algorithms to choose an AP with the strongest signal.

A potential issue could arise for audience members positioned in the middle between two access points with similar signal quality. In this case, a mobile device's Wi-Fi adapter could switch intermittently between the two access points. The switching process might take several seconds, during which users' interaction with the network would be interrupted. This would

create issues for any continuous streaming (audio/video) and cause disruptions in the user experience.

WebSocket and SSE clients should be able to reconnect when mobile devices rejoin the network. Therefore, the ZScore server would synchronise the mobile device's state upon reconnection. Depending on the current content in the mobile device's web browser, users might not notice the AP switching at all. However, the web front end and server state update logic should be designed with this problem in mind.

A.5.2.1 Wireless Mesh vs Multiple APs

Another option for the scenario above would be to deploy a wireless mesh solution. Mesh systems automatically negotiate client connectivity between wireless nodes, providing a superior network client switching logic. However, it is important to consider that with a wireless mesh system, all network traffic is transmitted wirelessly between the mesh nodes, which could create a throughput bottleneck on the router's entry point.

On the other hand, network traffic coming from multiple wired APs can be split through hardware switches, allowing for much higher aggregated bandwidth. Therefore, multiple APs are a recommended solution where high network throughput is required.

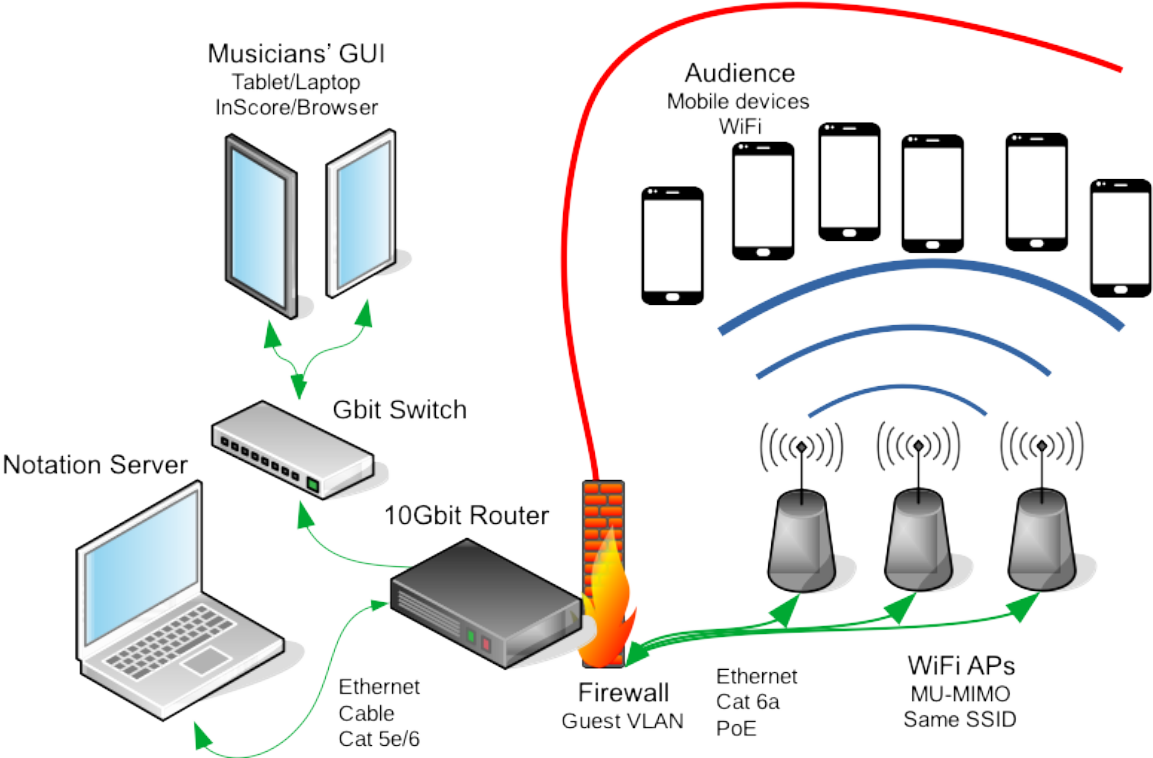


Figure A.8: System design for 500 Wi-Fi users

A.5.3 System Design For ~500 To ~1000 Wi-Fi Users

Assuming that the required bandwidth for 100 users is 200 Mbps, it becomes evident that a 1 Gbit router would not be able to cope with the typical network traffic as the number of connected Wi-Fi users approaches 500. In this case, an enterprise grade 10 Gbit router or a router providing independent Gigabit Ethernet ports (such as Mikrotik CCR1009-7G-1C-1S+PC) is required (Figure A.8). Ordinarily, enterprise grade routers typically do not provide any extra functionality, such as a built-in wireless AP. As a result, the complexity and cost of the system design would increase when incorporating additional access points to cater to the increased number of users.

A.5.4 System Design For 1000+ Wi-Fi Users

Tests have shown that the ZScore server can successfully handle up to 1500 connected users. However, once the user number goes beyond this limit, the ZScore server becomes the bottleneck, impacting performance. To address such a high load, the ZScore server's design needs to be modified. A possible solution is to run web servers in separate processes, and possibly on multiple hosts, to distribute the load (Figure A.9). Implementing a load balancing proxy in front of the web server instances can help marshal client connections efficiently. However, it is important to validate the WebSocket functionality when using any connection proxying, as it could impact the real-time communication between the clients and the server.

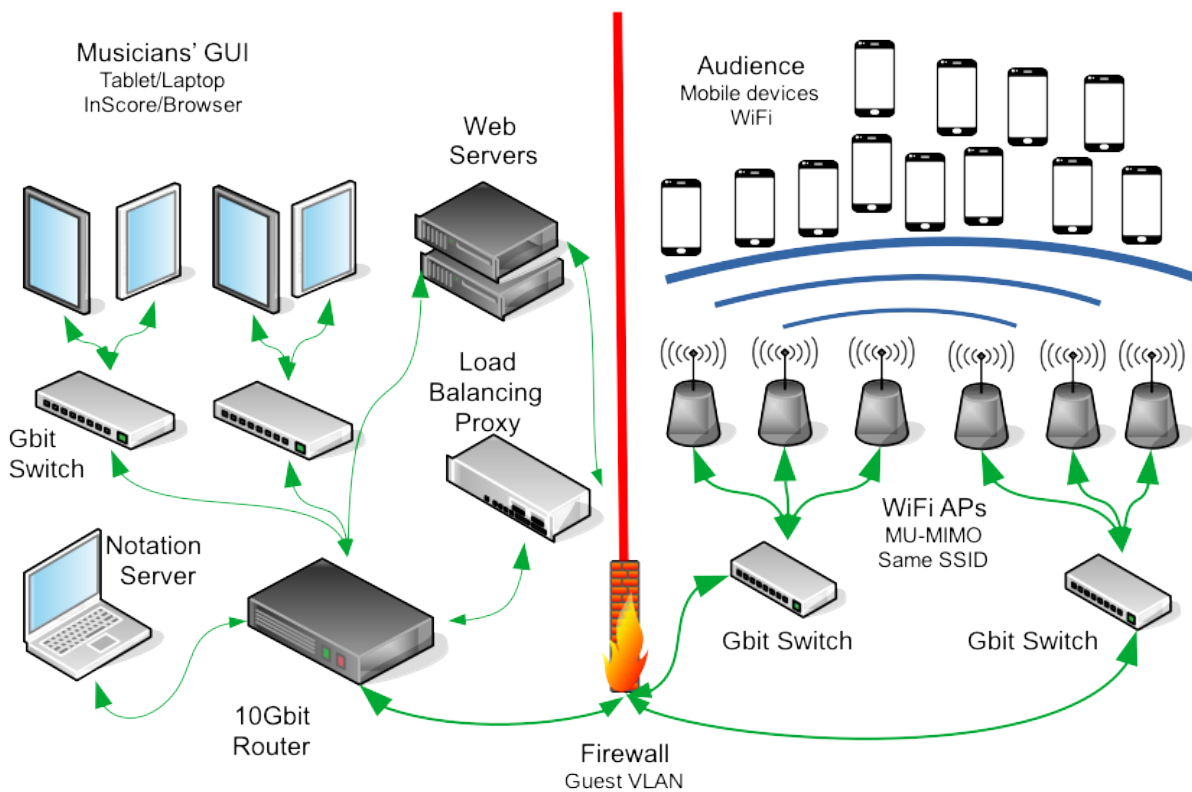


Figure A.9: System design for 1000+ Wi-Fi users

If the number of users exceeds 2000, then relying on a single 10 Gbit router would not be sufficient to handle the load, and a more comprehensive solution with multiple routers connected via optical cables is required. At that point the cost and complexity of the system deployment increase significantly.

A.6 Detailed Test Results

Test Id	Client Type	Connection type	Firewall	Client No	Latency			Percentile		
					Avg	Min	Max	90th	95th	99th
1	WebSocket	Wired	No	200	10	0	476	19	40	420
1	WebSocket	Wired	No	200	9	0	471	15	39	316
1	WebSocket	Wired	No	200	8	0	471	29	39	188
1	WebSocket	Wired	No	200	7	0	464	13	37	195
1	WebSocket	Wired	No	200	8	0	574	13	35	145
AVG					8	0	491	18	38	253

Table A.1: Test Case 1: Undertow WebSocket, Wired, No Firewall, 200 Clients

Test Id	Client Type	Connection type	Firewall	Client No	Latency			Percentile		
					Avg	Min	Max	90th	95th	99th
2	WebSocket	Wired	No	100	8	0	461	6	14	308
2	WebSocket	Wired	No	100	11	0	462	6	105	308
2	WebSocket	Wired	No	100	8	0	472	4	13	408
2	WebSocket	Wired	No	100	15	0	464	5	154	408
2	WebSocket	Wired	No	100	6	0	458	2	11	158
AVG					10	0	463	5	59	318

Table A.2: Test Case 2: Undertow WebSocket, Wired, No Firewall, 100 Clients

Test Id	Client Type	Connection type	Firewall	Client No	Latency			Percentile		
					Avg	Min	Max	90th	95th	99th
3	WebSocket	Wired	Yes	200	48	0	1044	260	401	726
3	WebSocket	Wired	Yes	200	39	0	841	141	365	460
3	WebSocket	Wired	Yes	200	39	0	960	50	361	570
3	WebSocket	Wired	Yes	200	39	0	791	41	363	692
3	WebSocket	Wired	Yes	200	35	0	702	77	339	508
AVG					40	0	868	114	366	591

Table A.3: Test Case 3: Undertow WebSocket, Wired, With Firewall, 200 Clients

Test Id	Client Type	Connection type	Firewall	Client No	Latency			Percentile		
					Avg	Min	Max	90th	95th	99th
4	WebSocket	Wired	Yes	100	30	0	626	24	305	391
4	WebSocket	Wired	Yes	100	24	0	390	26	268	355
4	WebSocket	Wired	Yes	100	30	0	676	27	303	370
4	WebSocket	Wired	Yes	100	29	0	518	63	269	379
4	WebSocket	Wired	Yes	100	26	0	390	20	293	377
AVG					28	0	520	32	288	374

Table A.4: Test Case 4: Undertow WebSocket, Wired, With Firewall, 100 Clients

Test Id	Client Type	Connection type	Firewall	Client No	Latency			Percentile		
					Avg	Min	Max	90th	95th	99th
5	WebSocket	Wireless	Yes	200	54	0	979	162	527	727
5	WebSocket	Wireless	Yes	200	60	0	1755	352	493	864
5	WebSocket	Wireless	Yes	200	54	0	1610	176	512	734
5	WebSocket	Wireless	Yes	200	49	0	1285	164	443	711
5	WebSocket	Wireless	Yes	200	55	0	1232	181	481	836
AVG					54	0	1372	207	491	774

Table A.5: Test Case 5: Undertow WebSocket, Wireless, With Firewall, 200 Clients

Test Id	Client Type	Connection type	Firewall	Client No	Latency			Percentile		
					Avg	Min	Max	90th	95th	99th
6	WebSocket	Wireless	Yes	100	46	0	745	133	413	582
6	WebSocket	Wireless	Yes	100	36	0	621	139	348	472
6	WebSocket	Wireless	Yes	100	41	0	823	135	431	478
6	WebSocket	Wireless	Yes	100	33	0	678	104	338	436
6	WebSocket	Wireless	Yes	100	41	0	514	193	388	482
AVG					39	0	676	141	384	490

Table A.6: Test Case 6: Undertow WebSocket, Wireless, With Firewall, 100 Clients

Test Id	Client Type	Connection type	Firewall	Client No	Latency			Percentile		
					Avg	Min	Max	90th	95th	99th
7	WebSocket	Wireless	No	200	55	0	1666	154	539	844
7	WebSocket	Wireless	No	200	56	0	1808	131	520	946
7	WebSocket	Wireless	No	200	64	0	1310	166	559	963
7	WebSocket	Wireless	No	200	60	0	1387	201	506	1002
7	WebSocket	Wireless	No	200	64	0	1653	204	520	953
AVG					60	0	1565	171	529	942

Table A.7: Test Case 7: Undertow WebSocket, Wireless, No Firewall, 200 Clients

Test Id	Client Type	Connection type	Firewall	Client No	Latency			Percentile		
					Avg	Min	Max	90th	95th	99th
8	WebSocket	Wireless	No	100	47	0	962	134	451	804
8	WebSocket	Wireless	No	100	37	0	1014	94	336	603
8	WebSocket	Wireless	No	100	46	0	881	137	464	868
8	WebSocket	Wireless	No	100	37	0	882	72	424	502
8	WebSocket	Wireless	No	100	29	0	831	117	315	461
AVG					39	0	914	111	398	648

Table A.8: Test Case 8: Undertow WebSocket, Wireless, No Firewall, 100 Clients

Test Id	Client Type	Connection type	Firewall	Client No	Latency			Percentile		
					Avg	Min	Max	90th	95th	99th
9	HTTP	Wired	No	200	109	69	233	164	186	189
9	HTTP	Wired	No	200	109	69	229	167	196	229
9	HTTP	Wired	No	200	102	60	234	141	168	232
9	HTTP	Wired	No	200	118	63	459	148	268	311
9	HTTP	Wired	No	200	112	64	260	175	188	259
				AVG	110	65	283	159	201	244

Table A.9: Test Case 9: Undertow HTTP, Wired, No Firewall, 200 Clients

Test Id	Client Type	Connection type	Firewall	Client No	Latency			Percentile		
					Avg	Min	Max	90th	95th	99th
10	HTTP	Wired	Yes	200	1025	182	4722	2095	3045	4675
10	HTTP	Wired	Yes	200	974	201	3678	2062	2788	3359
10	HTTP	Wired	Yes	200	982	190	3786	2173	2514	3592
10	HTTP	Wired	Yes	200	1271	187	6506	2541	4390	5993
10	HTTP	Wired	Yes	200	942	182	3553	2008	2841	3549
				AVG	1039	188	4449	2176	3116	4234

Table A.10: Test Case 10: Undertow HTTP, Wired, With Firewall, 200 Clients

Test Id	Client Type	Connection type	Firewall	Client No	Latency			Percentile		
					Avg	Min	Max	90th	95th	99th
11	HTTP	Wireless	Yes	100	1335	273	4739	2657	3335	4739
11	HTTP	Wireless	Yes	100	2357	270	6209	4650	5151	6209
11	HTTP	Wireless	Yes	100	1134	251	3372	2155	2730	3372
11	HTTP	Wireless	Yes	100	1465	271	4492	2808	3108	4492
11	HTTP	Wireless	Yes	100	2073	322	9220	4521	6420	9220
				AVG	1673	277	5606	3358	4149	5606

Table A.11: Test Case 11: Undertow HTTP, Wireless, With Firewall, 100 Clients

Test Id	Client Type	Connection type	Firewall	Client No	Latency			Percentile		
					Avg	Min	Max	90th	95th	99th
12	HTTP	Wireless	Yes	50	2190	318	5719	3935	5054	5719
12	HTTP	Wireless	Yes	50	895	274	2543	1662	1770	2543
12	HTTP	Wireless	Yes	50	1070	280	2129	1882	1919	2129
12	HTTP	Wireless	Yes	50	1570	304	5810	2952	3328	5810
12	HTTP	Wireless	Yes	50	3105	270	7162	5600	6292	7162
				AVG	1766	289	4673	3206	3673	4673

Table A.12: Test Case 12: Undertow HTTP, Wireless, With Firewall, 50 Clients

Test Id	Client Type	Connection type	Firewall	Client No	Latency			Percentile		
					Avg	Min	Max	90th	95th	99th
13	HTTP	Wireless	No	100	2637	382	6299	4714	5348	6299
13	HTTP	Wireless	No	100	1000	241	3901	2091	2805	3901
13	HTTP	Wireless	No	100	4279	606	8522	6970	8154	8522
13	HTTP	Wireless	No	100	3051	258	8278	5514	6535	8278
13	HTTP	Wireless	No	100	2751	284	8623	4785	5553	8623
				AVG	2744	354	7125	4815	5679	7125

Table A.13: Test Case 13: Undertow HTTP, Wireless, No Firewall, 100 Clients

Test Id	Client Type	Connection type	Firewall	Client No	Latency			Percentile		
					Avg	Min	Max	90th	95th	99th
14	HTTP	Wireless	No	50	2112	257	5598	4869	4987	5598
14	HTTP	Wireless	No	50	1018	321	3014	1857	2338	3014
14	HTTP	Wireless	No	50	701	244	2987	1372	1576	2987
14	HTTP	Wireless	No	50	684	255	2892	1215	1321	2892
14	HTTP	Wireless	No	50	1014	271	5114	1814	1821	5114
AVG					1106	270	3921	2225	2409	3921

Table A.14: Test Case 14: Undertow HTTP, Wireless, No Firewall, 100 Clients

100 Users, Wired, No Firewall				Latency milliseconds				Percentile		
Label	Sample No	KO	Error %	Average	Min	Max	Median	90th	95th	99th
/htp-112	100	0	0.00%	3.46	2	6	3	4	4.95	5.99
/htp-113	100	0	0.00%	3.36	2	5	3	4	4	5
/htp-114	100	0	0.00%	3.32	2	5	3	4	4	4.99
/htp-115	100	0	0.00%	3.22	2	9	3	4	4	8.99
/htp-116	100	0	0.00%	2.97	2	4	3	4	4	4
/htp-117	100	0	0.00%	2.95	2	4	3	4	4	4
/htp-118	100	0	0.00%	2.93	2	5	3	4	4	4.99
/htp-119	100	0	0.00%	2.81	2	5	3	3	4	5
/htp-120	100	0	0.00%	2.76	2	5	3	4	4	4.99
/htp-121	100	0	0.00%	2.91	2	27	3	3	4	26.77
/htp-122	100	0	0.00%	2.71	2	8	3	3	4	7.96
/htp-123	100	0	0.00%	2.69	2	4	3	3	3	4
/htp-124	100	0	0.00%	2.63	2	4	3	3	4	4
/htp-125	100	0	0.00%	2.54	1	5	3	3	3	4.99
/htp-126	100	0	0.00%	2.63	1	12	2.5	3	3.95	11.93
/htp-127	100	0	0.00%	2.51	1	7	2	3	4	6.98
/htp-128	100	0	0.00%	2.54	1	6	2	3	3	6
/htp-129	100	0	0.00%	2.45	1	7	2	3	3	6.97
/htp-130	100	0	0.00%	2.51	1	4	2	3	3.95	4
/htp-131	100	0	0.00%	2.45	2	4	2	3	3	4
/htp-132	100	0	0.00%	2.46	2	3	2	3	3	3
/htp-133	100	0	0.00%	2.48	2	6	2	3	3.95	5.99
/htp-134	100	0	0.00%	2.4	2	5	2	3	3	5
Polling Avg	2300	0	0	2.77	2	7	3	3	4	7
/test.html-103	100	0	0.00%	19.52	14	62	19	23	24	62

Table A.15: Test Case 15: JMeter, Wired, No Firewall, 100 Clients

500 Users, Wired, No Firewall				Latency milliseconds				Percentile		
Label	Sample No	KO	Error %	Average	Min	Max	Median	90th	95th	99th
/http-112	500	0	0.00%	6.85	1	133	3	14	28.9	66.97
/http-113	500	0	0.00%	6.64	1	142	3	13	24.95	69.9
/http-114	500	0	0.00%	6.16	1	78	4	12	19.95	59.99
/http-115	500	0	0.00%	6.41	2	211	3	12	17	52.95
/http-116	500	0	0.00%	5.91	1	148	3	11	15.95	59.89
/http-117	500	0	0.00%	5.87	1	156	3	11	19.95	49.96
/http-118	500	0	0.00%	6.4	2	272	3	11	19	68.86
/http-119	500	0	0.00%	5.86	1	167	3	12	17	48
/http-120	500	0	0.00%	6.55	1	272	3	11	19.95	65.86
/http-121	500	0	0.00%	7.39	1	316	3	13	26.9	68.89
/http-122	500	0	0.00%	6.88	1	273	3	12	19.95	67.94
/http-123	500	0	0.00%	6.8	1	321	3	12	22	61.84
/http-124	500	0	0.00%	6.23	1	263	3	11.9	20.95	55.99
/http-125	500	0	0.00%	5.54	1	79	3	13	18	41.99
/http-126	500	0	0.00%	5.71	1	83	3	13	21	50.85
/http-127	500	0	0.00%	5.64	1	139	3	12	17	56.97
/http-128	500	0	0.00%	5.59	1	202	3	10	15	45.98
/http-129	500	0	0.00%	5.58	1	108	3	11	18	44
/http-130	500	0	0.00%	5.92	1	87	3	12	21.95	65.96
/http-131	500	0	0.00%	6.29	1	136	3	11	21	62.99
/http-132	500	0	0.00%	6.51	1	194	3	10.9	20.95	72.94
/http-133	500	0	0.00%	6.58	1	302	3	10	21	51.98
/http-134	500	0	0.00%	7.29	2	204	3	11	23.95	104.96
Polling Avg	11500	0	0	6.29	1	186	3	12	20	61
/test.html-103	500	0	0.00%	30.4	14	421	18	50	88	268

Table A.16: Test Case 16: JMeter, Wired, No Firewall, 500 Clients

1000 Users, Wired, No Firewall				Latency milliseconds				Percentile		
Label	Sample No	KO	Error %	Average	Min	Max	Median	90th	95th	99th
/http-112	1000	0	0.00%	173.1	11	557	161	264.5	335.9	452.96
/http-113	1000	0	0.00%	163.69	4	625	154	239	283.95	436.86
/http-114	1000	0	0.00%	161.56	20	848	150	233	286.8	411.94
/http-115	1000	0	0.00%	157.86	18	580	148	228	274.85	420.93
/http-116	1000	0	0.00%	155.36	37	575	144	229.9	275.9	403.97
/http-117	1000	0	0.00%	152.35	26	574	145	223	250.95	405.99
/http-118	1000	0	0.00%	154.8	34	611	145	227.9	267.95	409.97
/http-119	1000	0	0.00%	152.19	4	542	145	223	268.85	399.99
/http-120	1000	0	0.00%	154.82	5	621	147	227.9	289.5	440.88
/http-121	1000	0	0.00%	150.88	6	576	146	223	255.9	388.97
/http-122	1000	0	0.00%	153.04	5	576	148	220	265.75	410.96
/http-123	1000	0	0.00%	152.84	29	554	148	221.9	253.9	408.94
/http-124	1000	0	0.00%	151.98	14	633	147	220	270.95	418.8
/http-125	1000	0	0.00%	153.16	17	519	151	217	260.95	410.91
/http-126	1000	0	0.00%	147.35	3	598	146	211.9	245.95	395.95
/http-127	1000	0	0.00%	148.74	20	737	143.5	223	259.75	447.94
/http-128	1000	0	0.00%	141.63	5	555	141	209	241.9	384.99
/http-129	1000	0	0.00%	139.2	3	598	139	208	236	378.84
/http-130	1000	0	0.00%	138.32	3	438	141	207.8	236.85	351.92
/http-131	1000	0	0.00%	135.42	2	488	141	205.9	222.95	314.99
/http-132	1000	0	0.00%	140.73	2	512	144.5	214.9	249.9	402.98
/http-133	1000	0	0.00%	141.6	3	586	144	219.9	261.95	417.94
/http-134	1000	0	0.00%	141.19	2	605	144.5	218.9	253.9	400.99
Polling Avg	23000	0	0	150.51	12	587	146	222	263	405
/test.html-103	1000	0	0.00%	2584.36	76	5338	2382	4982	5041	5246

Table A.17: Test Case 17: JMeter, Wired, No Firewall, 1000 Clients

1500 Users, Wired, No Firewall				Latency milliseconds				Percentile		
Label	Sample No	KO	Error %	Average	Min	Max	Median	90th	95th	99th
/http-112	1500	11	0.73%	430.89	3	17031	186.5	361.7	583.45	8433.16
/http-113	1500	0	0.00%	275.61	14	16491	182	307	355.95	1316.99
/http-114	1500	0	0.00%	213.16	4	1697	171	323	374	1309.99
/http-115	1500	0	0.00%	213.58	4	1514	165	331.8	397	1307.99
/http-116	1500	0	0.00%	197.58	3	1711	155	302	378.7	1233.98
/http-117	1500	0	0.00%	170.9	4	1534	142	266.9	328	573.95
/http-118	1500	0	0.00%	172.51	3	1697	138	249	325.95	1232.99
/http-119	1500	0	0.00%	172.29	4	1710	133	251.8	324.95	1308
/http-120	1500	0	0.00%	153.7	3	1704	127	228	282	1219.98
/http-121	1500	0	0.00%	141.66	3	1315	124	219.9	274.9	407.9
/http-122	1500	0	0.00%	141.12	3	1322	122	211.9	264.85	453.34
/http-123	1500	0	0.00%	134.42	3	1518	120	201	248	339.95
/http-124	1500	0	0.00%	134.92	4	1309	121	201	242.95	369.97
/http-125	1500	0	0.00%	134.79	4	1307	123	197	230	396.82
/http-126	1500	0	0.00%	134.91	4	1307	124	202.9	236.95	354
/http-127	1500	0	0.00%	135.34	4	554	126	202.9	229.95	309
/http-128	1500	0	0.00%	139.91	5	1221	132	213	233	322.98
/http-129	1500	0	0.00%	144.49	3	1234	137	217	248.9	367.97
/http-130	1500	0	0.00%	151.26	3	719	143	223	258	470.99
/http-131	1500	0	0.00%	156.76	3	721	150	227	263	562.93
/http-132	1500	0	0.00%	166.42	2	803	155	251	306.95	636.97
/http-133	1500	0	0.00%	170.18	2	848	158	258	328.95	608.96
/http-134	1500	0	0.00%	173.64	2	847	162	268	342.9	607.99
Polling Avg	34500	11	0	176.52	4	2614	143	248	307	1050
/test.html-103	1500	43	2.87%	7226.27	1	18070	7418	13530	16493	17008

Table A.18: Test Case 18: JMeter, Wired, No Firewall, 1500 Clients

100 Users, Wireless, With Firewall				Latency milliseconds				Percentile		
Label	Sample No	KO	Error %	Average	Min	Max	Median	90th	95th	99th
/http-112	100	0	0.00%	25.36	6	453	13	51.5	64.7	450.3
/http-113	100	0	0.00%	53.17	6	445	18	154.7	227.45	444.92
/http-114	100	0	0.00%	53.6	9	474	19	129.5	334.55	473.8
/http-115	100	0	0.00%	53.67	11	555	19	125.7	285.25	554.29
/http-116	100	0	0.00%	50.73	8	501	20	153	212.55	499.55
/http-117	100	0	0.00%	38.35	7	452	20	53	82.75	451.21
/http-118	100	0	0.00%	33.38	7	207	21	68	143.85	207
/http-119	100	0	0.00%	44.72	6	467	20	82.5	193.8	466.79
/http-120	100	0	0.00%	53.61	8	446	23.5	144.5	181	445.38
/http-121	100	0	0.00%	57.02	9	705	21.5	128.4	305	702.96
/http-122	100	0	0.00%	38.16	8	683	22	57.5	101.6	678.71
/http-123	100	0	0.00%	35.37	8	390	21	56.6	114.1	388.1
/http-124	100	0	0.00%	41.18	9	374	21	94.5	147.6	372.22
/http-125	100	0	0.00%	61.4	9	718	25	113.5	355.5	715.42
/http-126	100	0	0.00%	42.33	7	372	30	67.9	103.75	371.26
/http-127	100	0	0.00%	44.04	7	459	22	60.6	120.65	458.74
/http-128	100	0	0.00%	51.71	9	622	20	102.9	236.25	620.59
/http-129	100	0	0.00%	50.17	6	517	21	88.9	219.4	516.51
/http-130	100	0	0.00%	50.38	8	675	23	74.9	213.95	673.05
/http-131	100	0	0.00%	45.14	8	505	20	58.9	191.35	504.96
/http-132	100	0	0.00%	39.39	7	637	20	49.7	88.8	635.11
/http-133	100	0	0.00%	33.98	5	333	20	56.6	99.15	331.42
/http-134	100	0	0.00%	49.71	7	640	19.5	120.6	246.3	637.48
Polling Avg	2300	0	0	45.50	8	506	21	91	186	504
/test.html-103	100	0	0.00%	573.08	121	3254	331	1157	2059	3252

Table A.19: Test Case 19: JMeter, Wireless, With Firewall, 100 Clients

200 Users, Wireless, With Firewall				Latency milliseconds				Percentile		
Label	Sample No	KO	Error %	Average	Min	Max	Median	90th	95th	99th
/http-112	200	0	0.00%	366.58	15	2502	193	723.9	1388.05	2311.04
/http-113	200	0	0.00%	393.04	15	4551	176	1008.1	1282.05	3128.61
/http-114	200	0	0.00%	364.12	18	1900	192.5	932	1122.5	1880.18
/http-115	200	0	0.00%	349.76	13	1766	203.5	756.6	1191.25	1669.01
/http-116	200	0	0.00%	366.02	13	1873	199.5	836.8	1171.4	1823.57
/http-117	200	0	0.00%	368.83	14	3254	171	951.4	1301.45	2351.16
/http-118	200	0	0.00%	377.87	12	2783	172.5	1019.8	1304.55	2761.02
/http-119	200	0	0.00%	350.49	12	2446	174	858.4	1003.95	2393.43
/http-120	200	0	0.00%	377.49	12	2074	208.5	960	1271.75	1864.42
/http-121	200	0	0.00%	362.61	10	3235	196.5	742.9	1102.3	2105.69
/http-122	200	0	0.00%	379.71	11	4271	193	878.1	1102.65	2955.25
/http-123	200	0	0.00%	359.36	10	2338	171	910.8	1259.1	2100.58
/http-124	200	0	0.00%	316.94	15	1716	159.5	683.9	1290.55	1696.66
/http-125	200	0	0.00%	366.63	11	2406	169.5	973.1	1204.3	2255.53
/http-126	200	0	0.00%	289.68	13	1879	148.5	646.3	1011.45	1646.47
/http-127	200	0	0.00%	283.97	21	1763	148.5	666.5	916.9	1221.24
/http-128	200	0	0.00%	277.5	15	1680	147.5	658.2	866.9	1510.51
/http-129	200	0	0.00%	250.61	13	1787	132.5	625.3	841.65	1733.67
/http-130	200	0	0.00%	239.68	12	1256	141.5	630.7	909.9	1089.9
/http-131	200	0	0.00%	280.4	12	3121	135.5	603.8	1023	2139.75
/http-132	200	0	0.00%	247.69	11	2349	124	655	848.15	1850.74
/http-133	200	0	0.00%	260.35	11	2066	126.5	618.9	1158.3	1802.99
/http-134	200	0	0.00%	245.23	13	1681	117.5	627.5	815.35	1660.39
Polling Avg	4600	0	0	324.98	13	2378	165	781	1104	1998
/test.html-103	200	0	0.00%	7712.39	460	18398	8300	13253	14035	16356

Table A.20: Test Case 20: JMeter, Wireless, With Firewall, 200 Clients

100 Users, Wireless, No Firewall				Latency milliseconds				Percentile		
Label	Sample No	KO	Error %	Average	Min	Max	Median	90th	95th	99th
/http-112	100	0	0.00%	48.96	8	549	22	122	231.6	548.61
/http-113	100	0	0.00%	91.41	8	1536	29	215.2	397.4	1528.32
/http-114	100	0	0.00%	92.35	7	1329	36	224.7	474.5	1321.85
/http-115	100	0	0.00%	80.63	7	959	38	203.8	518.95	955.29
/http-116	100	0	0.00%	64.08	7	699	32	91.5	429.05	697.83
/http-117	100	0	0.00%	64.47	7	871	33	113.2	246.55	869.02
/http-118	100	0	0.00%	100.63	7	1196	36	409.8	568.4	1192.94
/http-119	100	0	0.00%	71.46	7	1045	32	99.2	497.4	1042.65
/http-120	100	0	0.00%	81.24	7	968	33.5	192.1	527.7	965.69
/http-121	100	0	0.00%	60.7	6	544	32.5	75	374.45	543.82
/http-122	100	0	0.00%	75.33	7	1050	34.5	127	502.85	1046.66
/http-123	100	0	0.00%	80.61	5	984	34	188.7	524.55	979.95
/http-124	100	0	0.00%	74.83	4	729	33	168	505.7	727.84
/http-125	100	0	0.00%	94.36	4	681	36	371.4	551.9	680.16
/http-126	100	0	0.00%	52.98	4	534	34.5	89.2	185.55	532.9
/http-127	100	0	0.00%	76.04	4	967	36	145.1	366.8	965.39
/http-128	100	0	0.00%	59.86	4	968	35	83.7	148.85	963.87
/http-129	100	0	0.00%	56.99	5	559	33	71.2	220.1	558.95
/http-130	100	0	0.00%	79.95	4	1535	33	147.4	443.95	1530.34
/http-131	100	0	0.00%	92.06	5	967	33	190.2	531.75	966.94
/http-132	100	0	0.00%	66.5	4	636	34	119.7	380.15	635.08
/http-133	100	0	0.00%	57.79	5	541	33	87.4	368.75	540.65
/http-134	100	0	0.00%	65.35	5	571	32.5	131.4	424.85	570.47
Polling Avg	2300	0	0	73.42	6	888	33	159	410	885
/test.html-103	100	0	0.00%	807	240	2196	661	1460	1590	2196

Table A.21: Test Case 21: JMeter, Wireless, No Firewall, 100 Clients

200 Users, Wireless, No Firewall				Latency milliseconds				Percentile		
Label	Sample No	KO	Error %	Average	Min	Max	Median	90th	95th	99th
/http-112	200	0	0.00%	346.25	9	2225	103	1049.7	1411.45	2184.22
/http-113	200	0	0.00%	382.01	15	4290	119.5	965.9	1479.4	3460.38
/http-114	200	0	0.00%	378.89	18	2912	116.5	1098.5	1614.95	2704.52
/http-115	200	0	0.00%	335.24	19	3958	105	984	1353.2	2619.35
/http-116	200	0	0.00%	327.52	16	2820	105.5	1001.5	1275.55	2384.91
/http-117	200	0	0.00%	354.98	15	4894	100	1017.4	1227.05	2774.12
/http-118	200	0	0.00%	331.85	22	3192	122.5	938.1	1244.6	2116.89
/http-119	200	0	0.00%	438.9	14	4717	110	1178.1	1751.5	4664.63
/http-120	200	0	0.00%	352.19	15	3592	106.5	959	1212.8	2518.44
/http-121	200	0	0.00%	333.76	19	4177	103.5	963.8	1202.85	3108.28
/http-122	200	0	0.00%	308.24	13	3629	114	715.8	1101.8	3391.06
/http-123	200	0	0.00%	401.42	9	5894	109.5	1051.6	1667.7	5000.55
/http-124	200	0	0.00%	316.66	8	4482	95.5	793.6	1366.65	3611.46
/http-125	200	0	0.00%	288.8	7	2158	98	793.4	1119.95	1822.7
/http-126	200	0	0.00%	319	6	5891	102	878.2	1123.25	2247.85
/http-127	200	0	0.00%	290.3	8	2341	92	711.1	1309.4	2159.25
/http-128	200	0	0.00%	259.7	9	3941	91	591.5	1098.3	3381.09
/http-129	200	0	0.00%	299.42	6	3170	100.5	977.7	1173.4	2329.67
/http-130	200	0	0.00%	272.29	5	3056	94.5	696	1094.95	2174.54
/http-131	200	0	0.00%	265.19	6	3016	84	678.9	1249.4	2524.31
/http-132	200	0	0.00%	243.72	5	3769	86.5	716.8	1046.65	2057.21
/http-133	200	0	0.00%	247.42	5	2898	89.5	611.5	1087.75	2843.65
/http-134	200	0	0.00%	186.91	4	3111	81	557.8	582.8	1956.43
Polling Avg	4600	0	0	316.55	11	3658	101	867	1252	2784
/test.html-103	200	0	0.00%	5697	267	14238	4675	11099	12383	13315

Table A.22: Test Case 22: JMeter, Wireless, No Firewall, 200 Clients

Appendix B

Participants' Feedback

B.1 Musician Feedback Form

How satisfied were you with the workshop overall?

	1	2	3	4	5	
Not very	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	It was great

How satisfied were you with the system stability and performance?

Didn't work at all	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Worked very well
--------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	------------------

How easy, or difficult did you find the notation in <score>?

Too easy	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Impossible to perform
----------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	-----------------------

How satisfied were you with the dynamic notation overlays?

Impossible to read	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Very clear
--------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	------------

How useful did you find the interactive features?

Useless (didn't use it)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Very useful
-------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	-------------

How satisfying did you find performing with the amplified digital audio?

Didn't work at all	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Blended perfectly
--------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	-------------------

How satisfying did you find the audience participation elements?

It was a distraction	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Worked very well
----------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	------------------

What system or notation improvements would be the most beneficial in the future?

Any additional comments regarding the workshop?

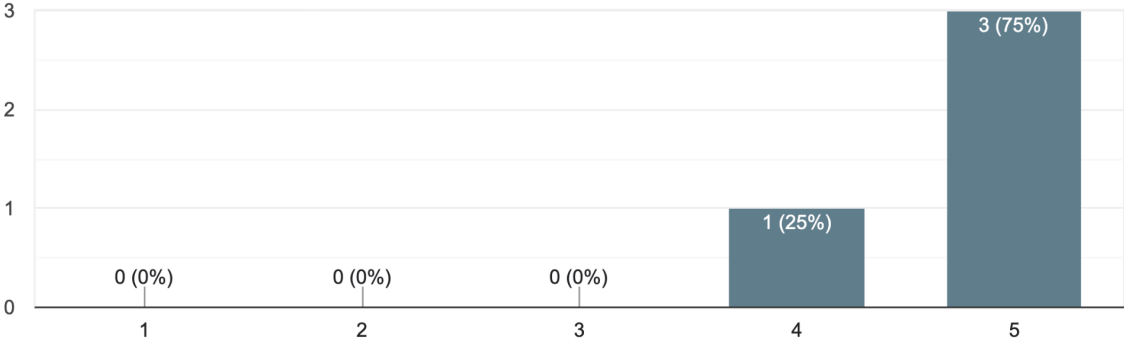
Figure B.1: Musicians' Feedback Form

B.2 Musicians' Feedback

B.2.1 Union Rose Workshop 23 Feb 2022

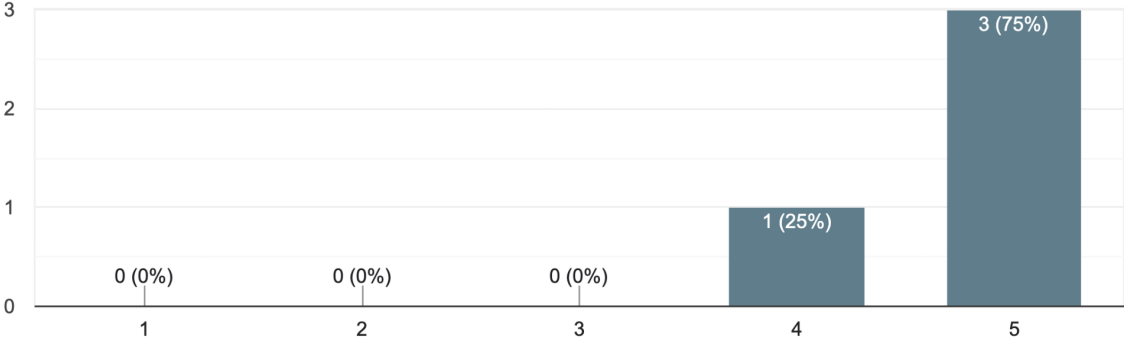
How satisfied were you with the workshop overall?

4 responses



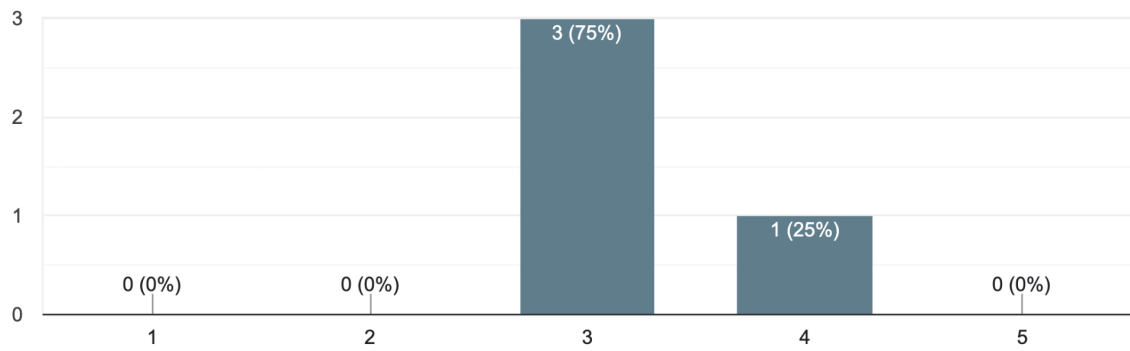
How satisfied were you with the system stability and performance?

4 responses



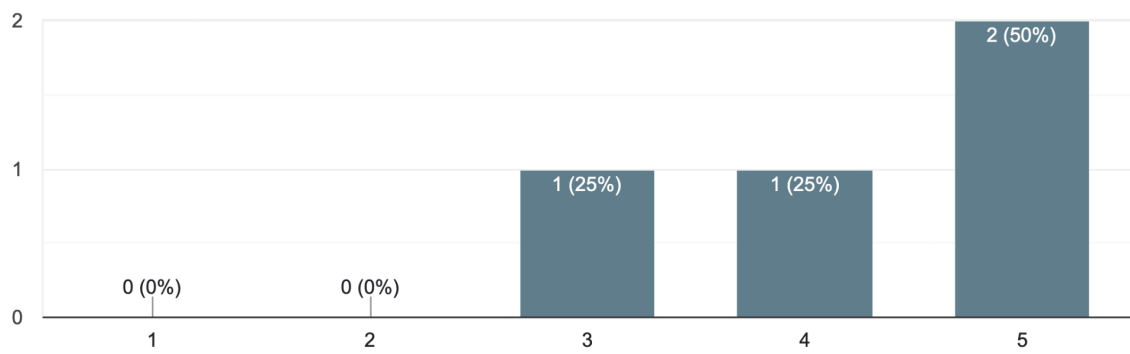
How easy, or difficult did you find the notation in 'Union Rose'?

4 responses



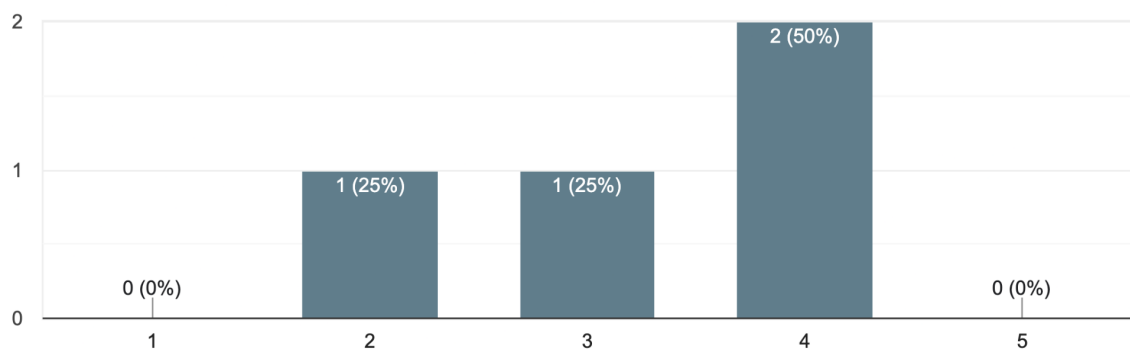
How satisfied were you with the dynamic notation overlays?

4 responses



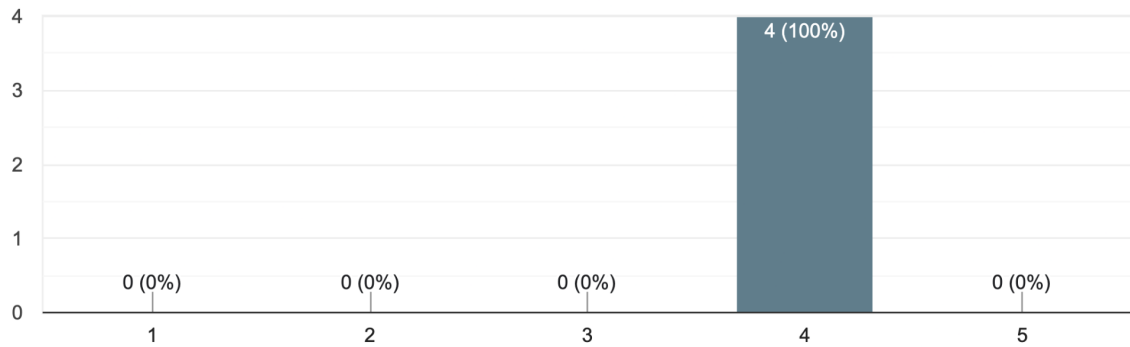
How useful did you find the interactive instrument opt in/out feature?

4 responses



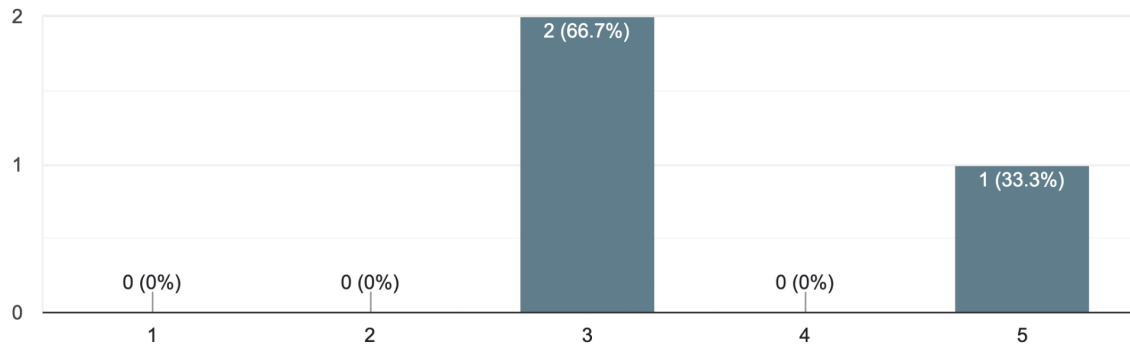
How satisfying did you find performing with the amplified digital audio?

4 responses



How satisfying did you find the audience participation elements?

3 responses



What system or notation improvements would be the most beneficial in the future? 3 responses

Having a beat's rest before the start of a next line when the music needs a sfz or solid start. Q3) difficult but very "performable". Q4) once in a while it was too quick to jump to the next line so I messed the first few notes. Q5) I used it rarely. Q7) wasn't particularly aware of the effects

Probably longer duration / larger scale. Q5) wanted more!

Indicate where any gesture must be in unison in the quartet (in the parts)

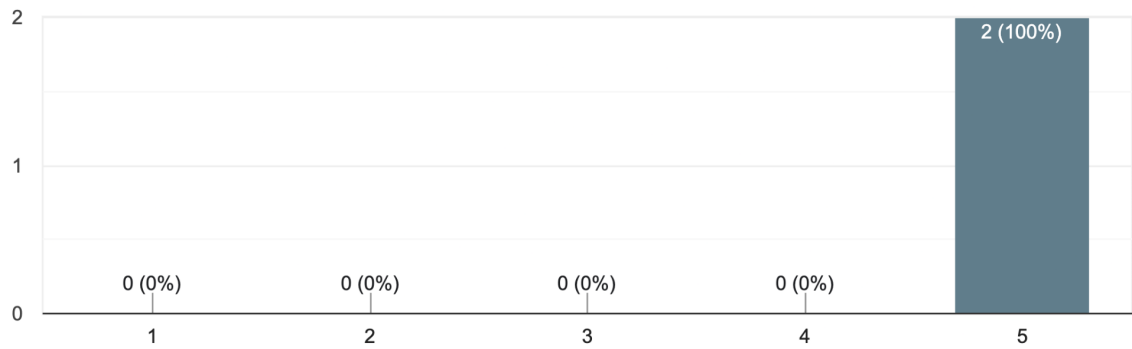
Great, thank you for involving us!!!

I'd be interested in people (not the composer) having more control e/g/ pitch, who plays etc.

B.2.2 *Socket Dialogues* Workshop 9 Jan 2023

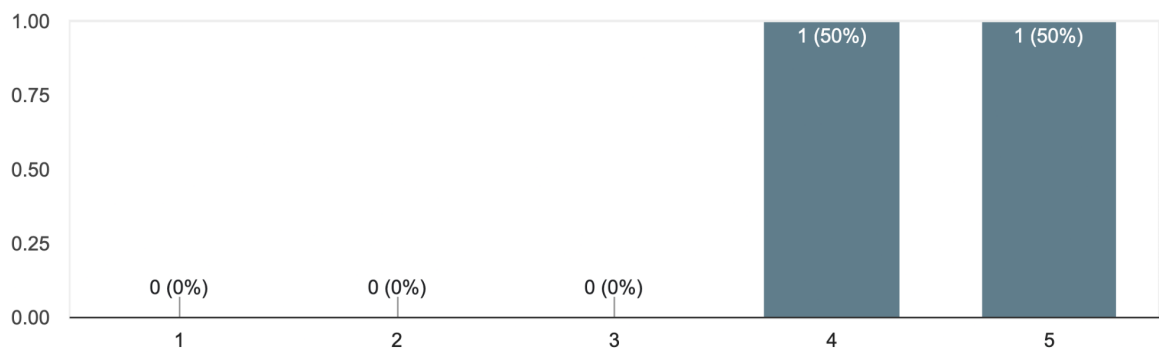
How satisfied were you with the workshop overall?

2 responses



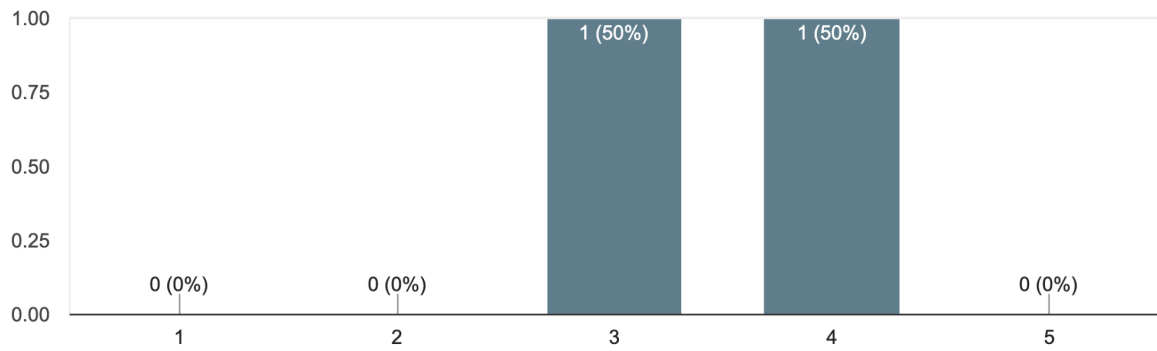
How satisfied were you with the system stability and performance?

2 responses



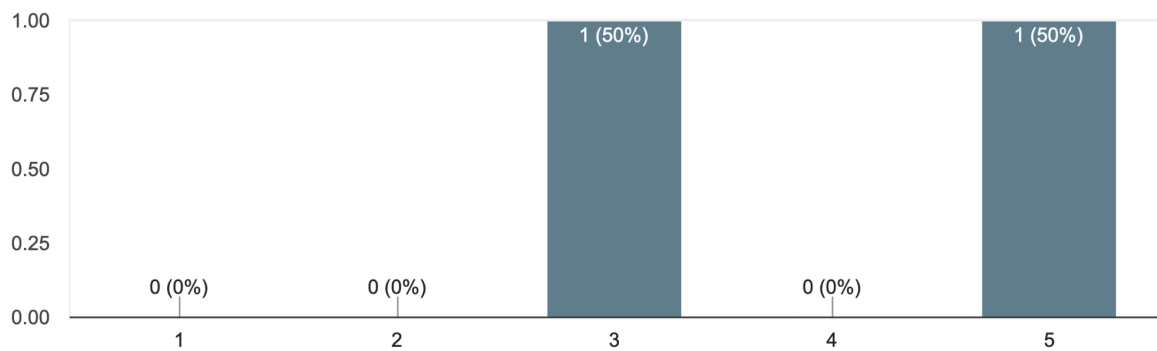
How easy, or difficult did you find the notation in 'Socket Dialogues'?

2 responses



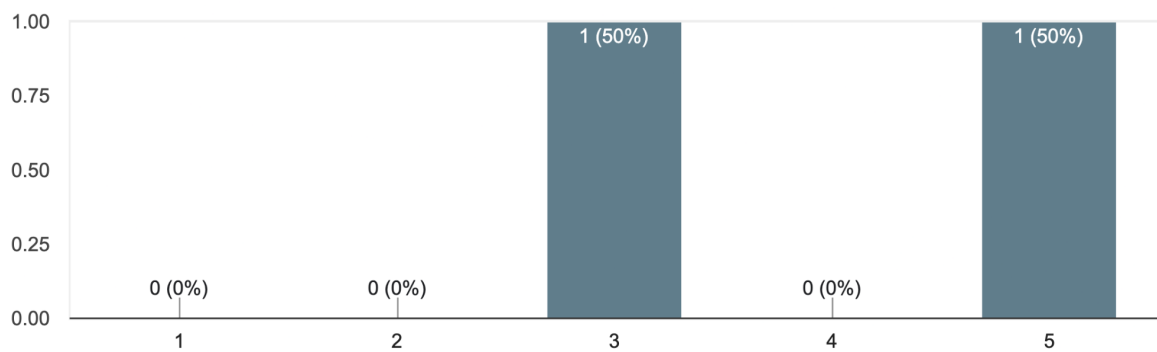
How satisfied were you with the dynamic notation overlays?

2 responses



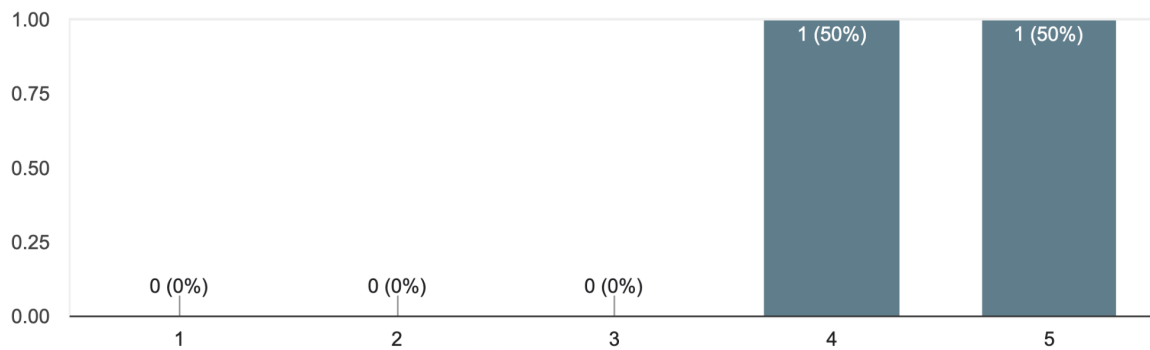
How useful did you find the interactive features?

2 responses



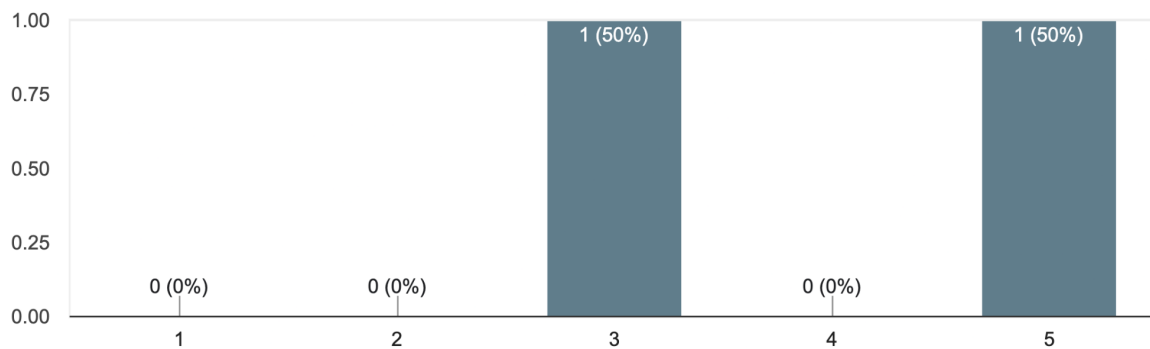
How satisfying did you find performing with the amplified digital audio?

2 responses



How satisfying did you find the audience participation elements?

2 responses



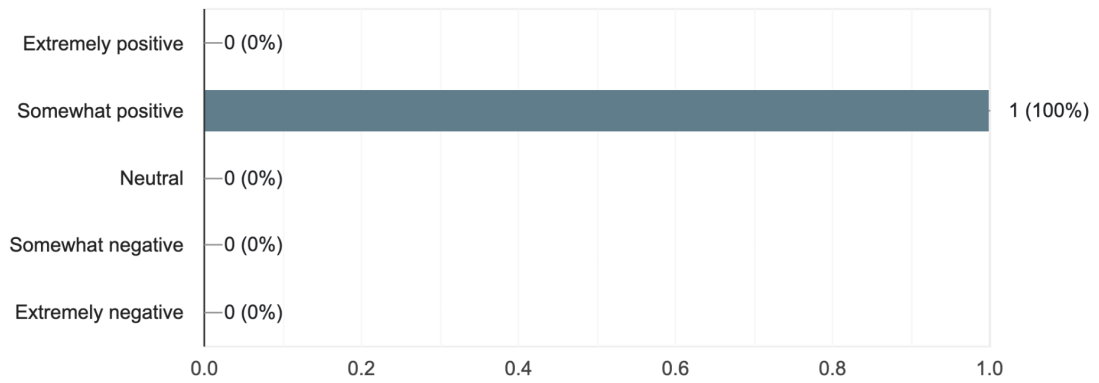
What system or notation improvements would be the most beneficial in the future? 1 response

Notations in classical staff system (especially in "melody") although I learned the melody in the end.

B.2.3 *Vexilla* Musician Feedback 1 May 2018

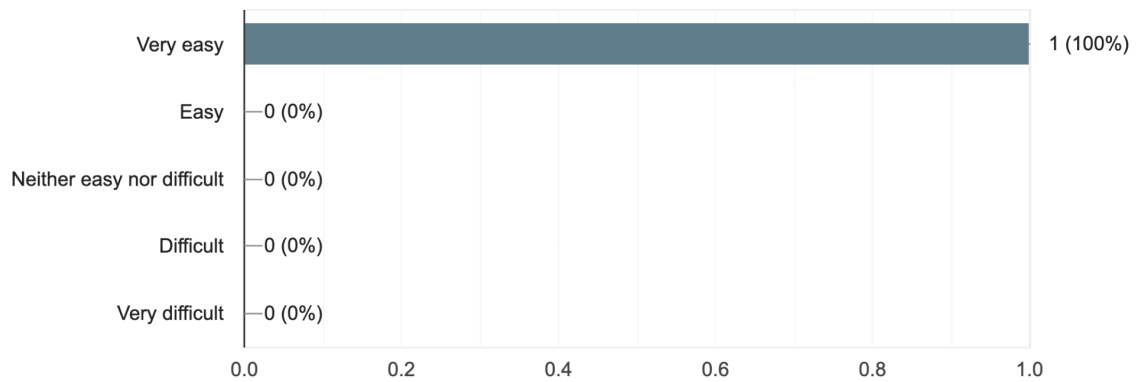
Overall, how was your ZScore experience?

1 response



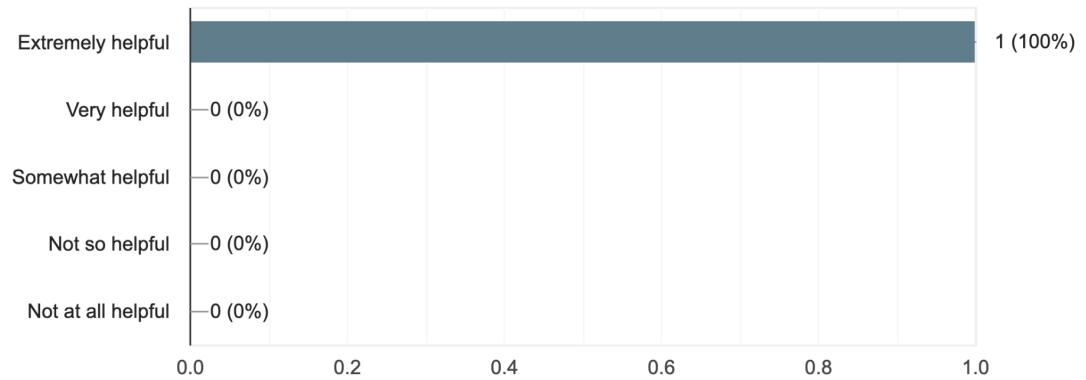
How easy, or how difficult did you find it to get used to ZScore's music notation display?

1 response



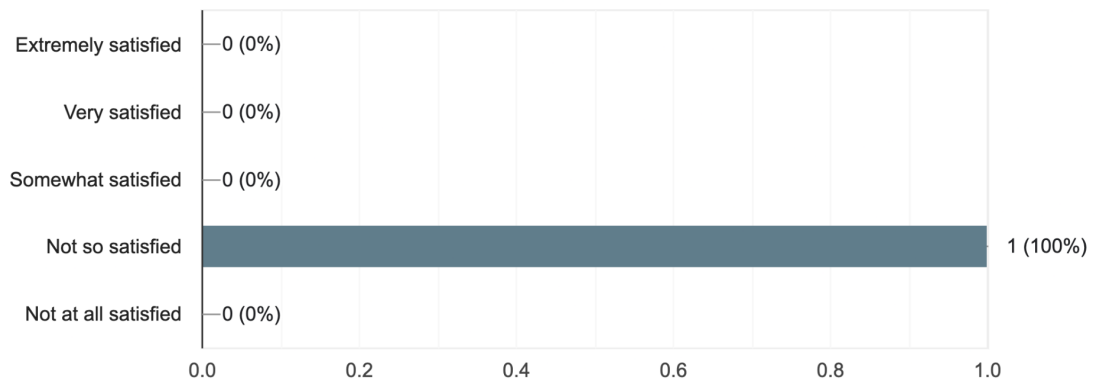
How helpful did you find the current position and tempo indicators?

1 response



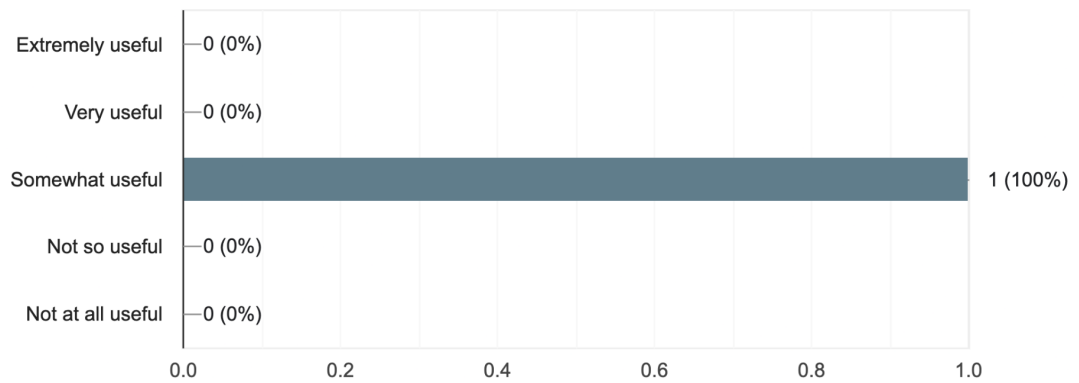
How satisfied are you with the notation layout in ZScore?

1 response



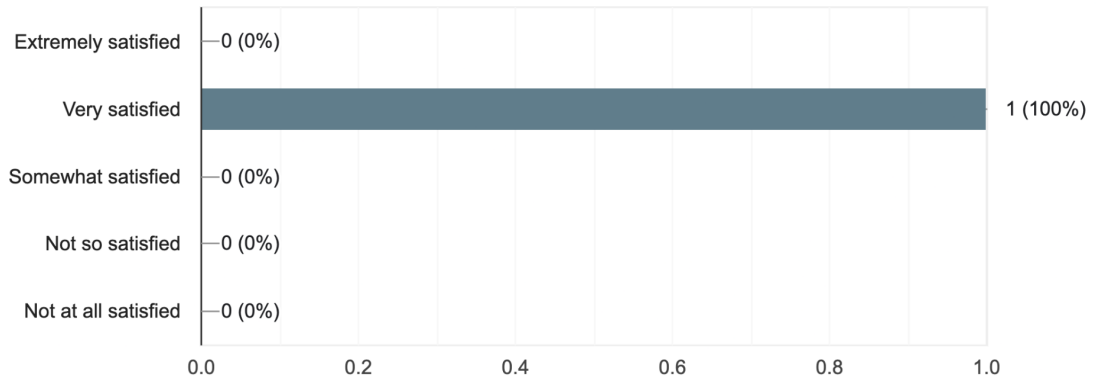
How useful did you find ZScore real-time notation distribution and network features?

1 response



How satisfied are you with the stability and performance of ZScore?

1 response



Please list the most negative aspects of ZScore¹ response

The spacing of the clarinet stave made it rather difficult to read the rhythms. I'm also not sure why it had to be networked - if you'd given us a score we could have just as easily played this as chamber music. But the real problem (and this is true of any animated score) is that you can't annotate. So any notes you give us or things we want to focus on we just have to remember. And this is just not how musicians work. If you could find a solution for this you would immediately make this kind of work 10x more viable.

Please list the most positive aspects of ZScore¹ response

Easy to know where we were at all times

B.3 Audience Feedback Form

How satisfied were you with the workshop overall?

1 2 3 4 5

Not very It was great

How engaged in the performance did you feel?

Zoned out Fully engrossed

How satisfied were you with the implementation of the mobile device score?

Didn't work at all Worked very well

Did the consequences of your actions match your expectations?

I couldn't comprehend what was going on I clearly understood the outcomes

Would you like to participate in another networked music performance?

Never again Definitely Yes

What were the most satisfying aspects of the performance? (Tick up to 3 that apply)

Music Mobile Score Musicians Other specify below:
 My participation Interactivity Digital audio

Any additional comments regarding the workshop?

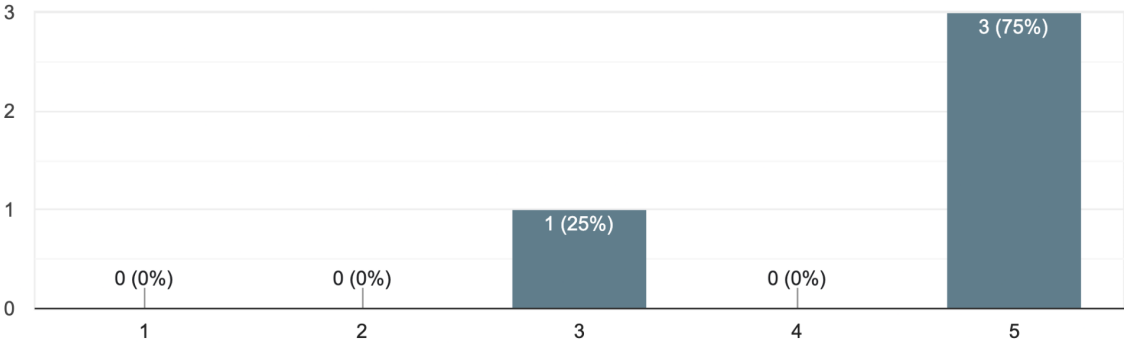
Figure B.2: Audience Feedback Form

B.4 Audience's Feedback

B.4.1 *Union Rose Workshop 23 Feb 2022*

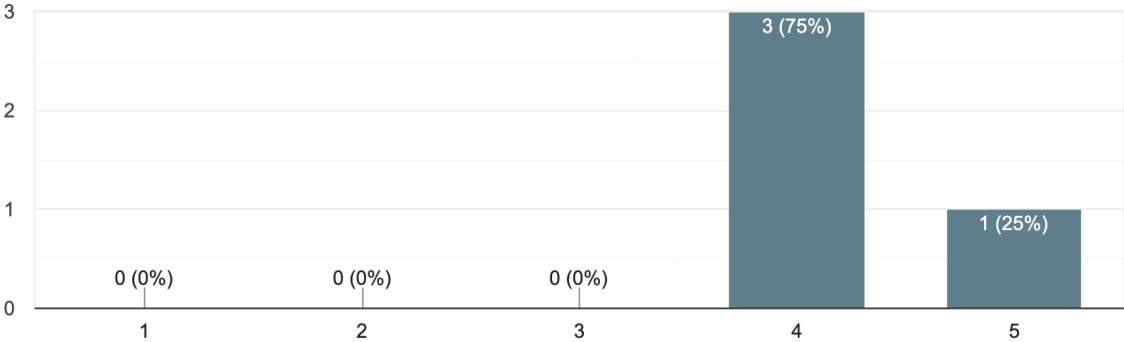
How satisfied were you with the workshop overall?

4 responses



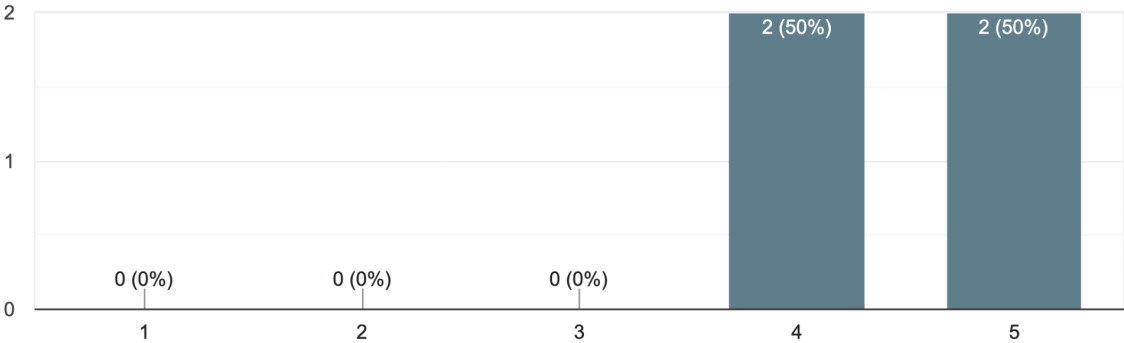
How engaged in the performance did you feel?

4 responses



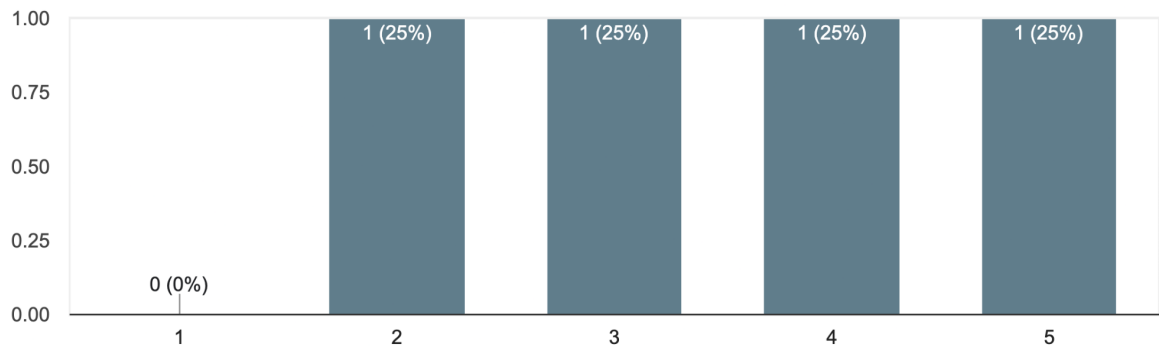
How satisfied were you with the mobile device score implementation?

4 responses



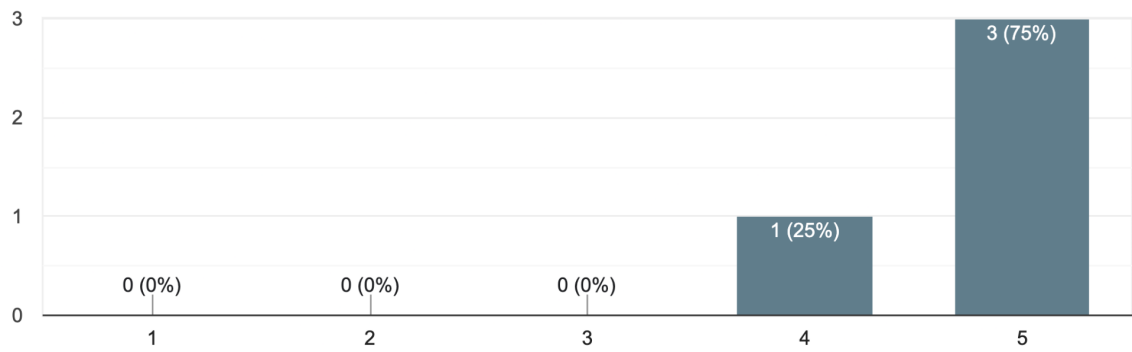
Did consequences of your actions match your expectations?

4 responses



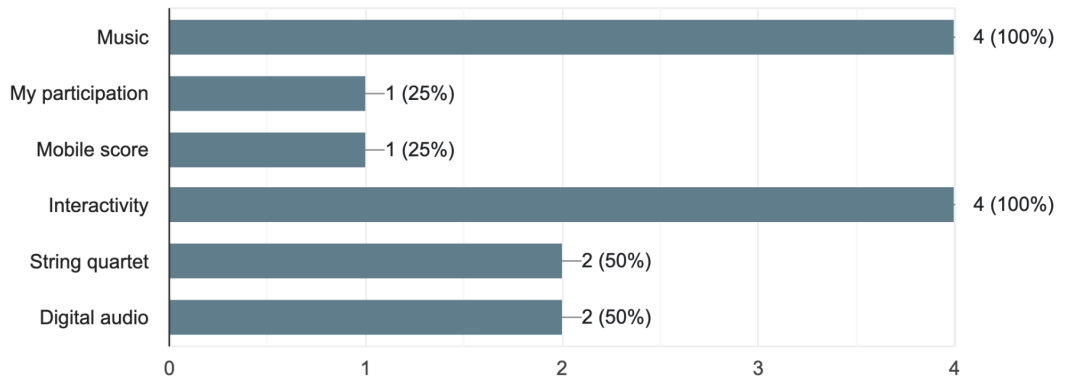
Would you like to participate in another networked music performance again?

4 responses



What were the most satisfying aspects of the performance?

4 responses



Any additional comments regarding the workshop? 3 responses

Nice Job! I'd like to see a bigger audience. We can talk more MZ

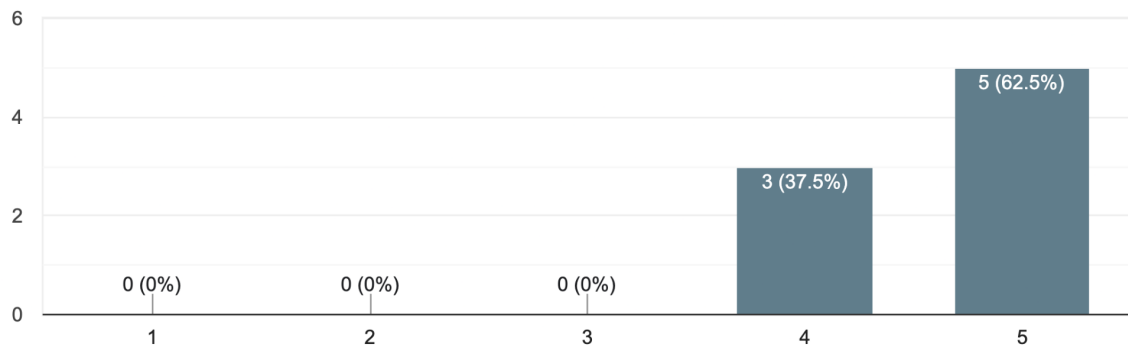
This was like therapy for me. Krystal

1. The music decided my different choice is quite similar, difficult to feel strong change (can feel a little bit, not too much). 2. Maybe the visualisation of the pattern behind the tile could also be changed by decision. (colour, shape, maybe use touch designer for program). 3. It's remind me the cooperation between jazz player, maybe they could also vote for their performance. 4. Maybe it's good for audience to see how the script change...

B.4.2 *Socket Dialogues* Workshop Audience Feedback 4 Jul 2022

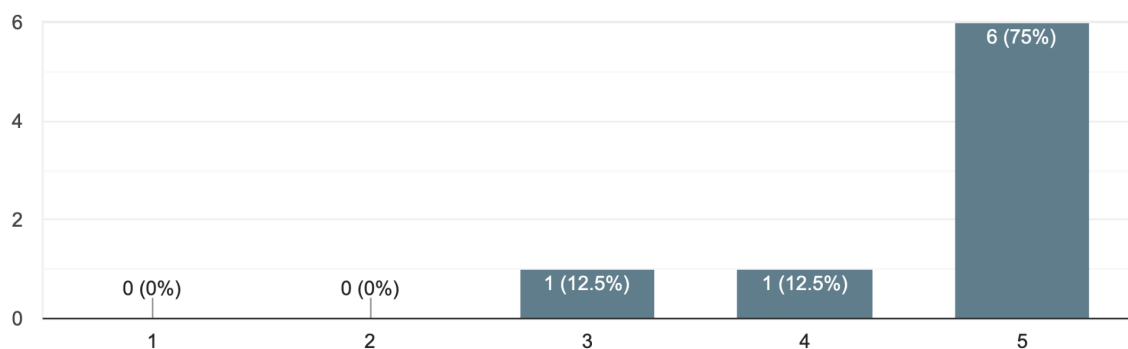
How satisfied were you with the workshop overall?

8 responses



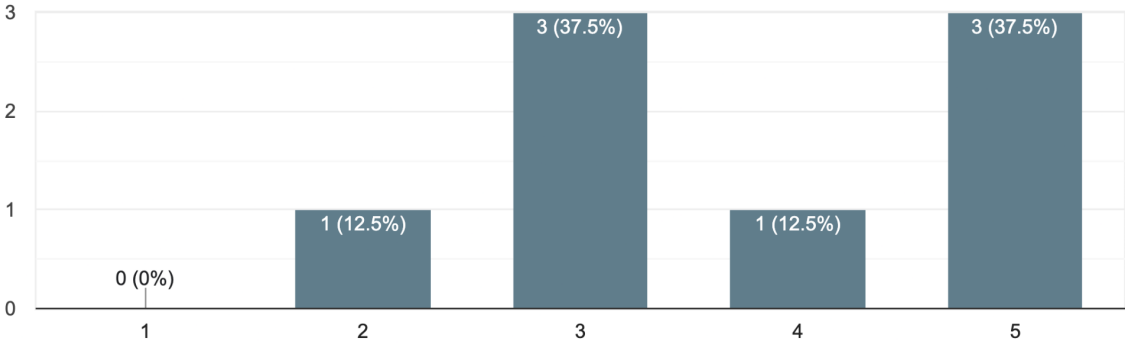
How engaged in the performance did you feel?

8 responses



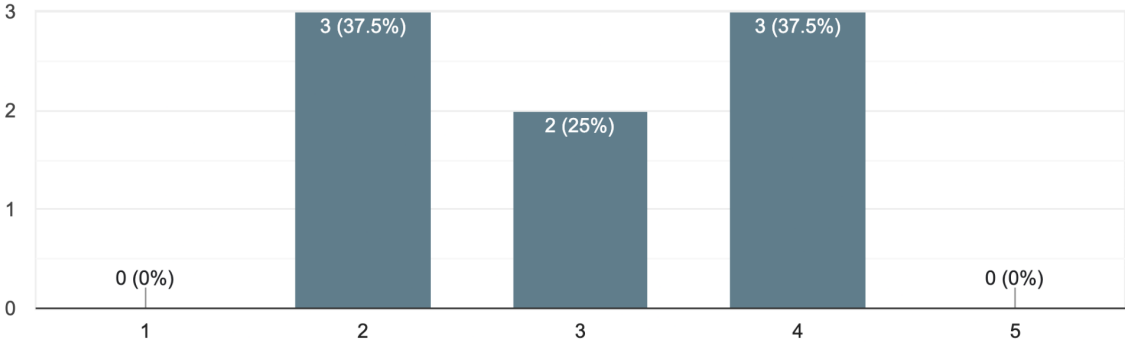
How satisfied were you with the implementation of the mobile device score ?

8 responses



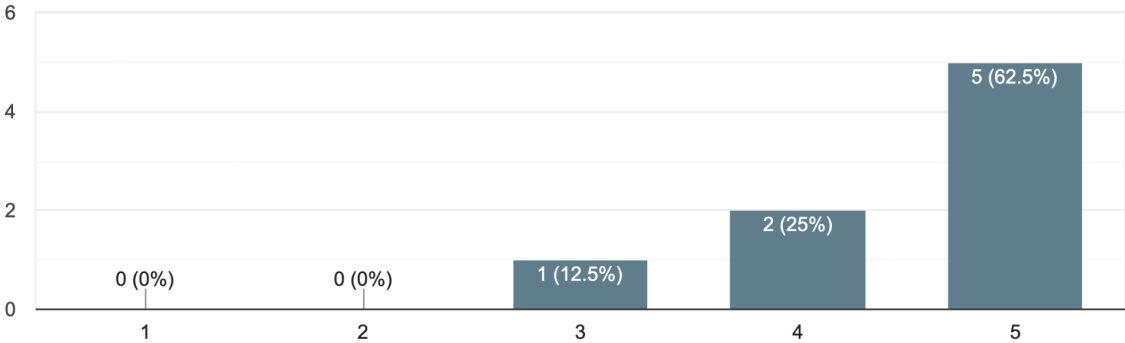
Did consequences of your actions match your expectations?

8 responses



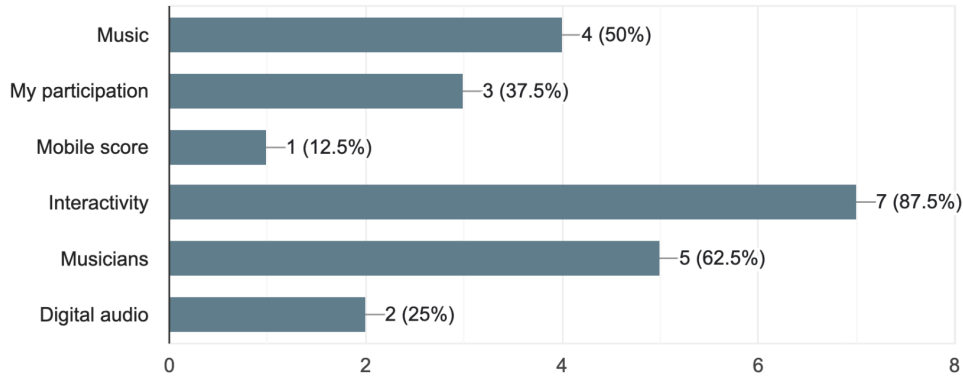
Would you like to participate in another networked music performance again?

8 responses



What were the most satisfying aspects of the performance?

8 responses



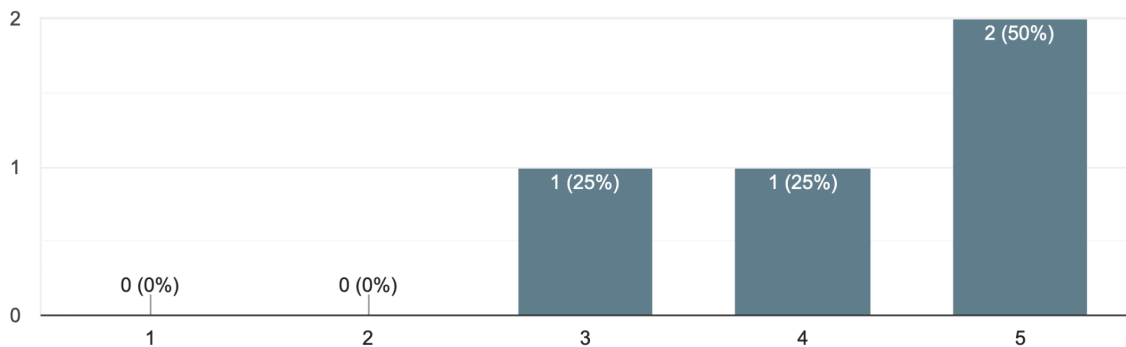
Any additional comments regarding the workshop? 5 responses

- It would be great to play more notes together as an audience. Maybe choose (rate) a certain segment of music that our phones play together, so we all feel like an instrument
- A great experience, I am happy to be involved in
- Is it possible for the audience to have autonomy on how much they have control on the influence? Personally I feel that satisfaction comes a lot from the revealed impact of interaction.
- Better relation between input & output would be great. Optionally, multi-modal input should be fun to try.
- I came late so I didn't quite understand what was going on but once I started interacting it was really fun and I didn't mind that much about understanding. Music & expression is what interests me the most and I was very impressed and satisfied with what I was hearing.

B.4.3 *Socket Dialogues* Workshop, Audience Feedback 9 Jan 2023

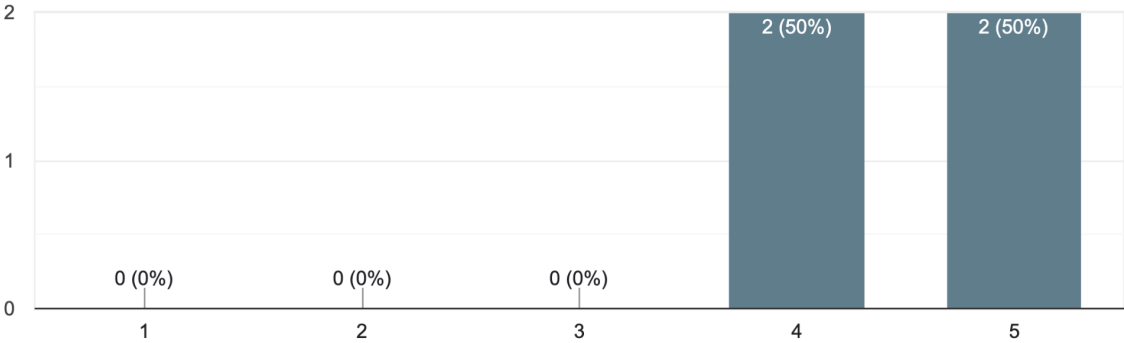
How satisfied were you with the workshop overall?

4 responses



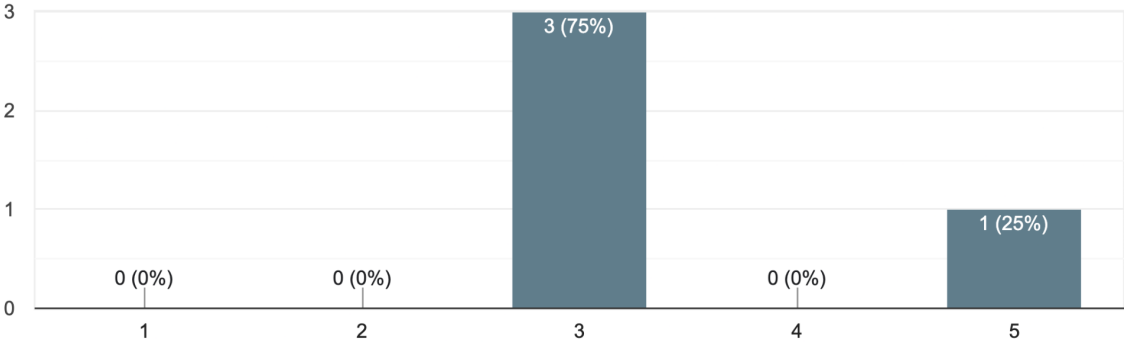
How engaged in the performance did you feel?

4 responses



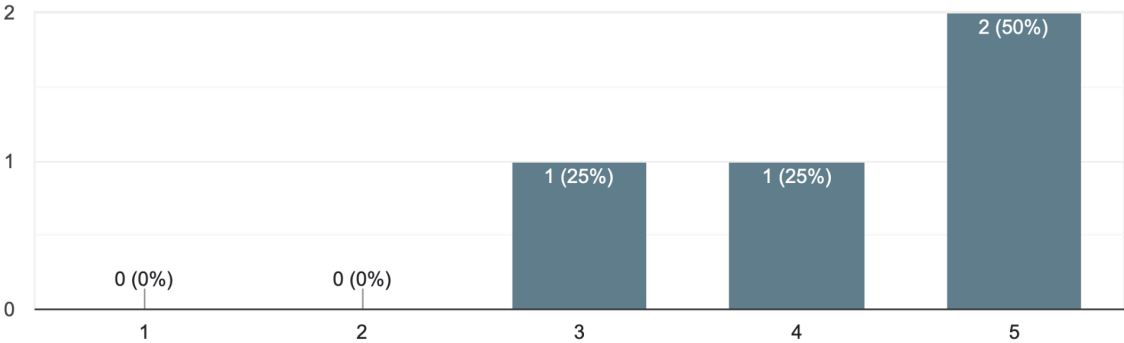
How satisfied were you with the implementation of the mobile device score ?

4 responses



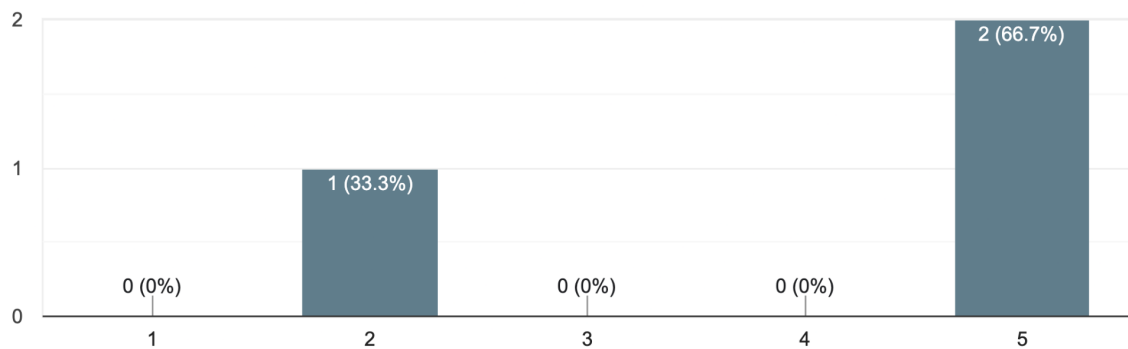
Did consequences of your actions match your expectations?

4 responses



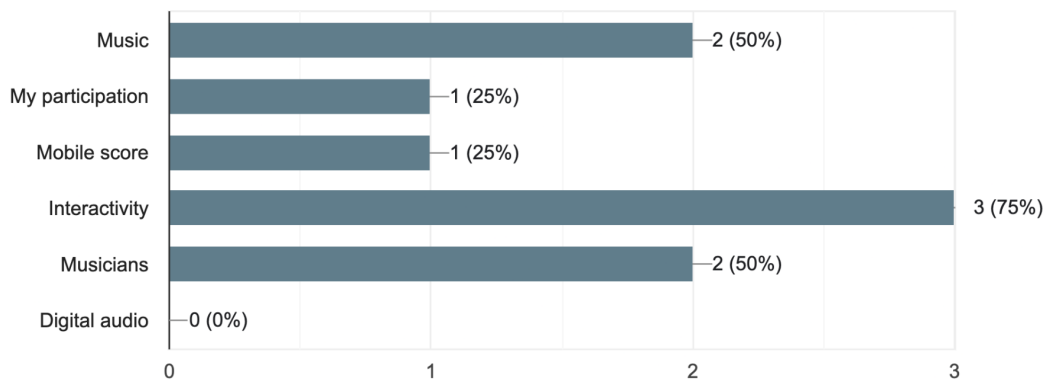
Would you like to participate in another networked music performance again?

3 responses



What were the most satisfying aspects of the performance?

4 responses



Any additional comments regarding the workshop? 3 responses

Found it uncomfortable to say 'dislike'. Enjoyed the note part the most & the noise on the improv - participation was clear here.

Interesting work. Would like to be in the know of future reseaches / developepment. Enjoyed interaction and collective participation --- increasing dynamic. The presentation - was informative though I probably expected a bit more performance -- (elements)

Regarding the sounds, I believe the sounds should tonally match the musicians. This occurred well during the "pitch" section.

B.4.4 Analysis Of The Audience Keyword Mention Frequency

Question: What were the most satisfying aspects of the performance?




Keyword	Frequency	Percent of total
Music	10	22%
Mobile score	3	7%
Interactivity	14	31%
Musicians	9	20%
My participation	5	11%
Digital audio	4	9%
TOTAL	45	

Figure B.3: Audience feedback keyword analysis

Appendix C

Flags Used In *Vexilla*

SFR Yugoslavia	
SR Slovenia	
SR Croatia	
SR Serbia	
SR Bosnia & Herzegovina	
SR Montenegro	

SR Macedonia	
EU	
Bosnia & Herzegovina (1998 — Now)	
UK	