1-1-2013

Designing, Implementing And Evaluating A Cross-Modal Sensory Substitution System For The Effective Communication Of The Emotional And Informative Aspects Of Music

Michael Pouris
Ryerson

DESIGNING, IMPLEMENTING AND EVALUATING A CROSS-MODAL SENSORY SUBSTITUTION SYSTEM FOR THE EFFECTIVE COMMUNICATION OF THE EMOTIONAL AND INFORMATIVE ASPECTS OF MUSIC

By

Michael Pouris

B.Sc. in Computer Science, Ryerson University, Toronto, Ontario, Canada 2011

A thesis presented to

Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Program of

Computer Science

Toronto, Ontario, Canada, 2013

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis including any required final revision, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

DESIGNING, IMPLEMENTING AND EVALUATING A CROSS-MODAL SENSORY SUBSTITUTION SYSTEM FOR THE EFFECTIVE COMMUNICATION OF THE EMOTIONAL AND INFORMATIVE ASPECTS OF MUSIC

Michael Pouris

M.Sc., Computer Science, Ryerson University, 2013

Abstract

Visualizations have existed for millennia as ways to communicate information. Visualizations are ubiquitous tools used every day to help navigate cities and aid in learning complex tasks. Tasks are made simpler when applying various visualization methods to large data sets to discover trends that are otherwise difficult to notice. More recently, music visualization systems have been created to convey music in the visual domain; however, they are not based on any psychological model of auditory and visual equivalents. This thesis discusses a music visualization system called MusicViz, which facilitates in the visual communication of the informative and entertainment aspect of music based on psychologically justified translation principals. MusicViz is combined with a vibro-feedback chair called the Emoti-Chair, which translates auditory music to vibrations along the user's back. The combined system is coined VITA (Visually Immersive and Tactile Animation). A usability evaluation of the VITA showed it is an enjoyable experience.

Acknowledgements

I would like to thank many people without whom I would not have been able to complete my thesis over the last two years at Ryerson University.

First and foremost, my supervisor, Dr. Deborah Fels, deserves my utmost most gratitude as without her guidance, help and support, this thesis would not have been possible. I was hired as a work-study student during my undergraduate degree and she was kind enough to guide me during my masters

In addition, I would like to thank my mother, Vasiliki Vittas, my step-father, James Cassells, and my grandmother –Yaya in Greek-, Fotoula Vittas, for providing me with emotional, moral and financial support as well as encouragement to produce a piece of work of which I may be proud. I would also like to thank my girlfriend, Nadina Michaels. She supported me in all my decisions and was kind enough to provide well-thought out suggestions and emotional support throughout performing my research. She was even kind enough to listen to the plethora of problems I encountered when performing the research, programming and even writing. I would also like to thank Arla Good for the initial work on the psychological models.

I would like to express my sincere gratitude to my close friends, Damyan Petkov, Mitko Tochev, Carly Harriss, Joseph Moscatiello, Kristian Ott and Hardip Jammu for providing much needed encouragement to work as hard as possible. I would like to also thank my close friends and colleagues at the IMDC lab, most importantly, Lisa Copeland and Leshanne pretty, without whom I would not have had the administrative power to acquire hardware, software or financial resources to complete my research. I would also like to thank Paul Church, Jorge Mori and Sai Cherukumilli that provided their overwhelming support, friendship and company during the long hours working in the lab and without them I probably would not have taken breaks to go to the gym or unwind playing StarCraft 2. Lastly, I would also like to thank all my friends that I was not able to mention as if I mentioned everyone that provided help and support, the list would be as long as this thesis. Also, I would like to thank colleagues within the IMDC at

Ryerson University, who were an integral part of creating a positive, collaborative, upbeat, out-going and inspiration work environment wherein I was able to perform this research.

Table of Contents

AUTHOR'S DECLARATION ii

Abstract iii

Acknowledgements iv

Table of Contents vi

List of Tables xiv

List of Figures xx

List of Equations xxv

List of Appendices xxvi

Table of Acronyms xxix

1 Introduction 1

1.1 Thesis Statement 4

1.2 Research Questions 5

1.3 Contributions of the Thesis 6

2 Literature Review 8

2.1 Introduction 8

2.2 Visualization 9

2.2.1 Visualization as Sensory Substitution 13

2.2.2 Music Visualization 13

2.2.2.1 Pragmatic Music Visualizations 14
2.2.2.2 Sublime Music Visualizations 26
2.2.2.3 Hybrid Music Visualizations 30
2.2.2.3.1.1 Psychological Models of Music 35
2.2.2.3.1.2 MusicViz's Psychological Model 38
2.3 Vibrotactile 40
2.3.1 Physiology and Frequency Theory 41
2.3.2 Vibration as Sensory Substitution 43
2.3.3 Haptic Sensory Substitution Devices 46
2.3.4 Thesis Outline 49
3 System Design and Evaluation Methodology 50
3.1 Overview 50
3.2 MusicViz Software 52
3.2.1 MIDI Standard and Java MIDI API 53
3.2.1.1 Note-On/Off Messages for Percussion and Instruments 54
3.2.1.2 Pitch-Bend Messages 55
3.2.2 Visual Structures in MusicViz 55
3.2.3 MIDI Sound Playback Structure 56
3.2.4 MusicViz Program Initialization and OpenGL Structure 57
3.2.5 MusicViz Playback and Visualization Architecture 59

3.2.5.1 Stage 1 60
3.2.5.2 Stage 2 60
3.2.5.2.1 Instrument Processing Thread 61
3.2.5.2.2 Beat Processing Thread 62
3.2.5.2.3 Pitch-Bend Processing Thread 62
3.2.5.3 Stage 3 62
3.2.5.3.1 Animating Instrument Messages 63
3.2.5.3.2 Animating Pitch-Bend Messages 64
3.2.5.3.3 Animating Percussion Messages 65
3.2.5.4 Stage 4 65
3.3 VITA: The Visually Immersive and Tactile Animation System 66
3.3.1 VITA Architecture 66
3.3.2 VITA Playback Scenarios 67
3.4 Methodology 67
3.4.1 Procedure 68
3.4.2 Research Questions 68
3.4.3 Participants' Demographics 69
3.4.4 Data Analysis 70
4 Evaluation 71
4.1 Results 71

4.1.1 Per-song Results 71
4.1.1.1 Levels of Enjoyability 71
4.1.1.1.1 Within Genres 71
4.1.1.1.2 Within Genres Chi-Sqaure 71
4.1.1.1.3 Between Hearing and Deaf Groups 72
4.1.1.1.4 Between Conditions 73
4.1.1.1.5 Within Genres and Between Hearing and Deaf Groups 73
4.1.1.1.6 Within Genres and Between Conditions 73
4.1.1.2 Levels of Continuous Emotions 73
4.1.1.2.1 Within Genres 73
4.1.1.2.2 Between Deaf and Hearing Groups 74
4.1.1.2.3 Between Conditions 74
4.1.1.2.4 Within Genres and Between Hearing and Deaf Groups 75
4.1.1.2.5 Within Genres and Between Conditions 75
4.1.1.3 Levels of Discrete Emotion Ratings 76
4.1.1.3.1 Within Genres 76
4.1.1.3.2 Between Hearing and Deaf Groups 81
4.1.1.3.3 Between Conditions 81
4.1.1.3.4 Within Genres and Between Hearing and Deaf Groups 82
4.1.1.3.5 Within Genres and Between Conditions 83

4.1.1.4 Levels of Distraction and Focus 83
4.1.1.4.1 Within Genres Chi-Square 83
4.1.1.5 Level of Agreement 85
4.1.1.5.1 Within Genres 86
4.1.1.5.2 Between Hearing and Deaf Groups 87
4.1.1.5.3 Between Conditions 87
4.1.1.5.4 Within Genres and between Hearing and Deaf Groups 88
4.1.1.5.5 Within Genres and Between Conditions 89
4.1.2 Post-study Results 89
4.1.2.1 Levels of Enjoyability 89
4.1.2.1.1 Chi-Square 89
4.1.2.1.2 Between Hearing and Deaf Groups 89
4.1.2.1.3 Between Conditions 90
4.1.2.2 Meaning of the Visual Constructs in the Visualization 90
4.1.2.2.1 Between Hearing and Deaf Groups 90
4.1.2.2.2 Between Conditions 92
4.1.2.3 Levels of Agreement 94
4.1.2.3.1 Chi-Square 94
4.1.2.3.2 Between Hearing and Deaf Groups 95
4.1.2.3.3 Between Conditions 95

4.1.2.4 NASA Task-Load Index 96
4.1.2.4.1 Mental Demand 96
4.1.2.4.2 Physical Demand 96
4.1.2.4.3 Temporal Demand 97
4.1.2.4.4 Performance 97
4.1.2.4.5 Effort 97
4.1.2.4.6 Frustration 98
4.2 Discussion 98
4.2.1 Per-song Discussion 98
4.2.1.1 Levels of Enjoyability 98
4.2.1.1.1 Within Genres 98
4.2.1.1.2 Between Deaf and Hearing Groups 100
4.2.1.2 Levels of Discrete and Continuous Emotions 101
4.2.1.2.1 Within Genres 101
4.2.1.2.2 Between Conditions 105
4.2.1.2.3 Within Genres and Between Deaf and Hearing Groups 111
4.2.1.2.4 Comparison between Discrete and Continuous Emotions 114
4.2.1.3 Levels of Distraction and Focus 115
4.2.1.3.1 Within Genres Chi-square 115
4.2.1.4 Levels of Agreement 116

4.2.1.4.1 Between Genres 116
4.2.1.4.2 Between Conditions 117
4.2.2 Post-study Discussion 120
4.2.2.1 Levels of Enjoyability 120
4.2.2.1.1 Chi-Square 120
4.2.2.2 Levels of Agreement 121
4.2.2.2.1 Chi-Square 121
4.2.2.3 Answers to Questions 121
4.2.2.3.1 Between Deaf and Hearing Groups 122
4.2.2.3.2 Between Conditions 125
4.3 Limitations 127
4.3.1 MusicViz's Technical Limitations 127
4.3.1.1 Dated 3D graphics and Lack of Complex Animations 127
4.3.1.2 Lack of Stereoscopic 3D 128
4.3.1.3 Lack Changeable Colour Palettes 128
4.3.2 Emoti-Chair's Limitations 128
4.3.3 Limited Number of Participants 129
4.3.4 Complexity of the Language in the Questionnaires 129
5 Conclusions and Future Work 130
5.1 Conclusions 130

5.2 Future Work 134

5.2.1 Research Suggestions 134

5.2.2 Technical Suggestions 135

6 Appendix 138

References 251

List of Tables

Table 1: MusicViz's audio to visual mapping 40

Table 2: MusicViz's audio to visual mapping 53

Table 3: Relevant MIDI messages to MusicViz [77] 53

Table 4: A note-on message targeting channel 11 and the three methods of specifying a note-off equivalent 55

Table 5: MusicViz's processing stages and associated classes 59

Table 6: Descriptive statistics and p-values for chi-square values relating to the levels of enjoyability for each genre (df=2) 72

Table 7: Descriptive statistics for the levels of enjoyability for each hearing status 72

Table 8: Descriptive Statistics for the levels of emotions between conditions 75

Table 9: Spearman's rho correlation between the ratings of happiness, sadness, anger and fear for classical music (N=20) 76

Table 10: Spearman's rho correlation between the ratings of happiness, sadness, anger and fear for country music (N=20) 77

Table 11: Spearman's rho correlation between the ratings of happiness, sadness, anger and fear for jazz music (N=20) 78

Table 12: Spearman's rho correlation between the discrete levels of happiness, sadness, anger and fear for pop music (N=20) 79

Table 13: Spearman's rho correlation between the ratings of happiness, sadness, anger and fear for RHRB music (N=20) 79

Table 14: Spearman's rho correlation between the ratings of happiness, sadness, anger and fear for rock music (N=20) 80

Table 15: Descriptive statistics for the levels of discrete emotions between conditions 81

Table 16: Descriptive statistics for the levels of anger for each genre and hearing status (N=20, 10 D, 10 H) 82

Table 17: Chi-Square results relating to the levels of distraction from the visualization for each genre (df=2) 84

Table 18: Chi-Square results relating to the levels of distraction from the Emoti-Chair for each genre (dF=2 and N=20) 84

Table 19: Chi-Square values results to the levels of focus for each genre..... 85

Table 20: Descriptive statistics for the levels of agreement with statement 5 for each genre (N=17)..... 86

Table 21: Descriptive statistics for the levels of agreement for statements 1 to 4 for each condition..... 88

Table 22: Descriptive statistics for the levels of agreement of each statement..... 95

Table 23: Descriptive Statistics for the NASA TLX between conditions..... 107

Table 24: Descriptive Statistics for the levels of anger for each hearing status and genre 112

Table 25: MIDI classes and interfaces in the Java 6 API [86] 201

Table 26: Data layout of a single vertex in graphics memory 205

Table 27: Memory structure of a pipe composed of *N* vertices in graphics memory 206

Table 28: MIDI percussion mapping [87]..... 210

Table 29: Descriptive statistics for the levels of enjoyability for each genre (N=19) 213

Table 30: Descriptive statistics for the levels of enjoyability for each condition 214

Table 31: Descripvie statistics for the levels of enjoyability within genres and between hearing status groups..... 214

Table 32: Descriptive statistics for the levels of enjoyability within genres and between conditions..... 215

Table 33: Descriptive statistics for the levels of valence for each genre (N=20) 216

Table 34: Descriptive statistics for the levels of arousal for each genre (N=20) 216

Table 35: Descriptive statistics for the levels of overbearingness for each genre (N=20)..... 216

Table 36: Spearman's rho correlation between the levels of valence, arousal and overbearingness for classical music (N=20, df= 18) 219

Table 37: Spearman's rho correlation between the levels of valence, arousal and overbearingness for country music (N=20, df=18) 219

Table 38: Spearman's rho correlation between the levels of valence, arousal and overbearingness for jazz music (N=20, df=18)..... 220

Table 39: Spearman's rho correlation between the levels of valence, arousal and overbearingness for pop music (N=20, df=18)..... 220

Table 40: Spearman's rho correlation between the levels of valence, arousal and overbearingness for RHRB music (N=20, df=18)..... 221

Table 41: Spearman's rho correlation between the levels of valence, arousal and overbearingness for rock music (N=20, df=18)..... 222

Table 42: Descriptive Statistics for the levels of emotions for each hearing status..... 222

Table 43: Descriptive statistics for the levels of valence for each genre and hearing status (N=20, 10 Deaf, 10 Hearing) 223

Table 44: Descriptive statistics for the levels of arousal for each genre and hearing status (N=20, 10 Deaf, 10 Hearing) 223

Table 45: Descriptive statistics for the levels of overbearingness for each genre and hearing status (N=20, 10 Deaf, 10 Hearing)..... 224

Table 46: Descriptive statistics for the levels of valence for each genre and condition 225

Table 47: Descriptive statistics for the levels of arousal for each genre and condition 226

Table 48: Descriptive statistics for the levels of overbearingness for each genre and condition..... 227

Table 49: Descriptive statistics for the discrete level of happiness for each genre (N=20)..... 228

Table 50: Descriptive statistics for the discrete level of sadness for each genre (N=20) 228

Table 51: Descriptive statistics for the discrete level of anger for each genre (N=20)..... 228

Table 52: Descriptive statistics for the discrete level of fear for each genre (N=20) 229

Table 53: Descriptive statistics for the levels of discrete emotions for each hearing status 229

Table 54: Descriptive statistics for the levels of happiness for each genre and hearing status (N=20, 10 Deaf, 10 Hearing)..... 230

Table 55: Descriptive statistics for the levels of sadness for each genre and hearing status (N=20, 10 Deaf, 10 Hearing) 231

Table 56: Descriptive statistics for the levels of fear for each genre and hearing status (N=20, 10 Deaf, 10 Hearing) 231

Table 57: Descriptive statistics for the discrete levels of happiness for each genre and condition 232

Table 58: Descriptive statistics for the discrete levels of sadness for each genre and condition..... 233

Table 59: Descriptive statistics for the discrete levels of anger for each genre and condition 234

Table 60: Descriptive statistics for the discrete levels of fear for each genre and condition..... 235

Table 61: Descriptive statistics for the levels of agreement with statement 1 for each genre (N=17)..... 236

Table 62: Descriptive statistics for the levels of agreement with statement 2 for each genre (N=17)..... 236

Table 63: Descriptive statistics for the levels of agreement with statement 3 for each genre (N=17)..... 236

Table 64: Descriptive statistics for the levels of agreement with statement 4 for each genre (N=17)..... 237

Table 65: Descriptive statistics for the levels of agreement with statement 6 for each genre (N=17)..... 237

Table 66: Descriptive statistics for the levels of agreement with statement 7 for each genre (N=17)..... 237

Table 67: Descriptive statistics for the levels of agreement with statements 1 to 3 for each hearing status 238

Table 68: Descriptive statistics for the levels of agreement with statement 4 to 7 for each hearing status 239

Table 69: Descriptive statistics for the levels of agreement for statements 5 to 7 for each condition..... 239

Table 70: Descriptive statistics for the levels of agreement with statement 1 for each genre and hearing status 240
Table 71: Descriptive statistics for the levels of agreement with statement 2 for each genre and hearing status 241
Table 72: Descriptive statistics for the levels of agreement with statement 3 for each genre and hearing status 241
Table 73: Descriptive statistics for the levels of agreement with statement 4 for each genre and hearing status 242
Table 74: Descriptive statistics for the levels of agreement with statement 5 for each genre and hearing status 243
Table 75: Descriptive statistics for the levels of agreement with statement 6 for each genre and hearing status 243
Table 76: Descriptive statistics for the levels of agreement with statement 7 for each genre and hearing status 244
Table 77: Descriptive statistics for the levels of agreement with the statement "the movement was on the screen was hard to watch" 245
Table 78: Descriptive statistics for the levels of agreement with the statement “the patterns and shapes on the screen were pleasant to view” 245
Table 79: Descriptive statistics for the levels of agreement with the statement "the vibrations helped me understand the visualization" 246
Table 80: Descriptive statistics for the levels of agreement with the statement “the visualization helped me understand the vibrations” 246
Table 81: Descriptive statistics for the levels of agreement with the statement: “the vibrations were too strong" 247

Table 82: Descriptive statistics for the levels of agreement with the statement "the vibrations up and down my back felt pleasant" 247

Table 83: Descriptive statistics for the levels of agreement with the statement "the pattern of vibrations on my back was confusing" 248

Table 84: Descriptive statistics for the levels of enjoyability for each condition 249

Table 85: Descriptive statistics for the levels of enjoyability for each hearing status 249

Table 86: Descriptive statistics for the levels of agreement of each statement for each hearing status 249

List of Figures

Figure 1: "The gamut of data-based visualization. a) Parallel Sets … show data about the people on the Titanic, and are readable and recognizable as a visualization; b) Ambient visualization … visualizing a bus schedule are readable but require more effort and are not readily recognizable as a visualization; c) Music visualization like MilkDrop … is also based on data, but not readable" [18] 11

Figure 2: Static orthogonal view of visual music [12] 15

Figure 3: Static 3D view of the music composition [12] 15

Figure 4: A final static visualization of music from MISUAL [23] 17

Figure 5: Creation of a similarity matrix from a music stream [24] 19

Figure 6: A similarity matrix of music for music categorization [24] 19

Figure 7: Beat spectrogram of a musical piece [24] 19

Figure 8: Beat histogram of a musical piece [24] 19

Figure 9: Timbregram using spectrograph [24] 20

Figure 10: Timbregram using striped boxes [24] 20

Figure 11: Timbregram of individual songs [24] 20

Figure 12: Real-time visual categorization of music [24] 20

Figure 13: Islands of Music Visualization [24] 21

Figure 14: BRASS visualization using Chernoff Faces [11] 24

Figure 15: Performance visualization [26] 25

Figure 16: ENP visualization output [28] 26

Figure 17: Visualization of symmetric harmonies [28] 26

Figure 18: Windows Media Player 28

Figure 19: PS3 Gaia visualization [33] 28

Figure 20: xBox Visualizer [34] 28

Figure 21: iTunes Magnetosphere 28
Figure 22: WinAmp 28
Figure 23: WinAmp 28
Figure 24: WinAmp 28
Figure 25: Atari Video Music [35] 28
Figure 26: Painted Music [32] 28
Figure 27: Windows Media Player Equalizer 28
Figure 28: Piano Roll [45] 32
Figure 29: Circle of Fifths [45] 32
Figure 30: Interval Types [45] 32
Figure 31: Interval Types [45] 32
Figure 32: Shapes [45] 32
Figure 33: Harmonic Staff [45] 32
Figure 34: Harmonic Compass [45] 32
Figure 35: Piano Roll Harmonic Compass [45] 32
Figure 36: Triads [45] 32
Figure 37: Part Sequence [45] 32
Figure 38: Part Motion [45] 32
Figure 39: Part Trajectory [45] 32
Figure 40: Motion Pixels of Music [3] 33
Figure 41: Dyad Visualization [46] 34
Figure 42: Circle of Thirds [46] 34
Figure 43: Current Emoti-Chair and actuator positions 47
Figure 44: 300 watt Pyramid Amplifier 48

Figure 45: Presonus Firepod 48
Figure 46: Haptic Chair [15] 49
Figure 47: General overview of VITA 51
Figure 48: MusicViz visualization (pipes for instruments and toroid objects for percussion) 56
Figure 49: Overview of how MusicViz's objects implement and intercept MIDI messages in real-time ..57
Figure 50: MIDI Player GUI Controls 58
Figure 51: Four processing stages of MusicViz's visualization algorithm 59
Figure 52: Pseudo-code and general structure of ConcurrentVisualizer.java's threads 63
Figure 53: VITA system architecture 66
Figure 54: VITA's Spatial Separation Scenarios 67
Figure 55: Frequency chart for participants' answers for each hearing status pertaining to the meaning of brightness in the visualization 91
Figure 56: Frequency chart for participants' answers for each hearing status pertaining to the meaning of the shapes in the visualization 91
Figure 57: Frequency chart for participants' answers for each hearing status pertaining to the meaning of the heights of the pipes in the visualization 92
Figure 58: Frequency chart for participants' answers for each condition pertaining to the meaning of brightness in the visualization 93
Figure 59: Frequency chart for participants' answers for each condition pertaining to the meaning of the shapes in the visualization 93
Figure 60: Frequency chart for participants' answers for each condition pertaining to the meaning of the heights of the pipes in the visualization 94
Figure 61: Descriptive statistics for the levels of discrete emotions for classical music (N=20) 102
Figure 62: Descriptive statistics for the levels of discrete emotions for country music (N=20) 103

Figure 63: Descriptive statistics for the levels of discrete emotions for jazz music (N=20) 103

Figure 64: Descriptive statistics for the levels of discrete emotions for pop music (N=20) 104

Figure 65: Descriptive statistics for the levels of discrete emotions for RHRB music (N=20) 104

Figure 66: Descriptive statistics for the levels of discrete emotions for rock music (N=20) 105

Figure 67: Descriptive Statistics for the NASA TLX between conditions 107

Figure 68: Discrete Levels of Emotions for each Condition 110

Figure 69: Descriptive Statistics for the levels of Anger between each Hearing Status and Genre 113

Figure 70: Overall Enjoyability of VITA regardless of genre, hearing status and condition 120

Figure 71: Assembling a pitch-bend message from two bytes 200

Figure 72: Creating a Java MIDI system for audio playback 203

Figure 73: Attaching a MIDI *Receiver* to a MIDI *Transmitter* 204

Figure 74: Calculating memory footprint for a single pipe 204

Figure 75: Allocating memory for a pipe using Figure 74 205

Figure 76: Vertices as stored in memory for a two-sided pipe 206

Figure 77: Visiting order of vertices to render pipe 206

Figure 78: Binding and drawing the buffer 207

Figure 79: MIDIPlayer class initialization 208

Figure 80: Original Receiver object running on the Sound Sequencer Thread 209

Figure 81: Sending MIDI messages from the *Java Sound Sequencer* to a separate thread for processing 209

Figure 82: Two OpenGL contexts [78] 212

Figure 83: Worker task architecture [78] 213

Figure 84: Descriptive statistics for the levels of valence for each genre 217

Figure 85: Descriptive statistics for the levels of arousal for each genre 218

Figure 86: Descriptive statistics for the levels of overbearingness for each genre 218

List of Equations

Equation 1: Calculate the number of cents *n* between note frequency *b* and *a* 41

Equation 2: Calculate the frequency of a note *b* that is *n* cents away from note *a* 41

Equation 3: Calculating the pipe's alpha value 64

Equation 4: Calculating the pipe's size 64

Equation 5: Calculating the y-axis position 64

Equation 6: Calculating a new pitch value 201

List of Appendices

6 Appendix 138

6.1 InstrumentReciever.java Source Code 138

6.2 BeatReceiver.java Source Code 142

6.3 PitchReceiver.java Source Code 145

6.4 ConcurrentVisualizer.java Source Code 148

6.5 Beat.java Source Code 160

6.6 ConcurrentPipe.java 164

6.7 GUI.java Source Code 179

6.8 Pre-study Questionnaire 189

6.9 After-each song Questionnaire 192

6.10 Post-study Questionnaire 195

6.11 Letter of Ethics Approval 199

6.12 General Midi Specification Description 200

6.12.1 Description and assembly of Pitch-Bend Messages 200

6.12.2 Java MIDI Playback Structure 201

6.13 MusicViz's Program Initialization and OpenGL Structure 204

6.14 MusicViz Playback and Visualization Architecture 208

6.14.1 Stage 1 208

6.14.2 Stage 2 208

6.14.2.1 MusicViz's Threading Model 208
6.14.2.2 Beat Processing Thread 210
6.14.3 Stage 3 211
6.14.3.1 Worker Task Architecture 211
6.15 Not Significant Per-song Results 213
6.15.1 Levels of Enjoyability 213
6.15.1.1 Within Genres 213
6.15.1.2 Between Conditions 214
6.15.1.3 Within Genres and between Hearing Status Groups 214
6.15.1.4 Within Genres and between Conditions 215
6.15.2 Levels of Continuous Emotions 216
6.15.2.1 Within Genres 216
6.15.2.2 Between Deaf and Hearing Groups 222
6.15.2.3 Within Genres and between Hearing and Deaf Groups 223
6.15.2.4 Within Genres and between Conditions 225
6.15.3 Levels of Discrete Emotion Ratings 228
6.15.3.1 Within Genres 228
6.15.3.2 Between Hearing and Deaf Groups 229
6.15.3.3 Within Genres and Between Hearing and Deaf Groups 230
6.15.3.4 Within Genres and between Conditions 232

6.15.4 Levels of Agreement 235
6.15.4.1 Within Genres 235
6.15.4.2 Between Hearing and Deaf Groups 238
6.15.4.3 Between Conditions 239
6.15.4.4 Within Genres and between Hearing and Deaf Groups 240
6.15.4.5 Within Genres and Between Conditions 245
6.16 Not Significant Post-study Results 249
6.16.1 Levels of Enjoyability 249
6.16.1.1 Between Conditions 249
6.16.1.2 Between Hearing and Deaf Groups 249
6.16.2 Levels of Agreement 249
6.16.2.1 Between Hearing and Deaf Groups 249

Table of Acronyms

ANOVA	Analysis of Variance
D	Deaf
H	Hearing
HOH	Hard-of-Hearing

1 Introduction

Music is a major art form that is present throughout all human cultures during recorded history [1]. Music acts as a transmitter of cultural information that spans time and space. Even though Ancient Greek and Chinese musical pieces are over two thousand years old, they still provide important insight to the emotional feelings and cultural states of their respective societies.

In modern Western Culture, music is everywhere and can be a powerful tool for entertainment, the propagation of culture, relaxation and even health improvement. It is heard in restaurants, spas, clubs, concerts, etc. With the advent of technological advancements such as MP3 players and digital compression algorithms, individuals carry hundreds and thousands of pre-recorded songs on MP3 players and phones (e.g., iPods, iPhones, BlackBerries and smart phones). Just as Ancient Greek and Chinese music provides insight into the collective culture over two thousand years ago, modern music conveys modern culture and provides a shared experience and knowledge space that spans cultural boundaries [2, 3]. However, individuals also listen to music for its entertainment value and its ability to invoke emotional reactions [3]. For example, individuals often experience flashbacks and physical sensations such as chills, blood pressure changes, quickening heart rate and tears [2-4]. Music is also shown to be a powerful anti-anxiety tool that may work better than medication [5].

[3] states that music is often not enjoyed as sound alone; it can be accompanied by visuals, such as facial expressions, body language and various special effects in television shows, live performances and movies in order to reinforce emotional information. Even though this is redundant information, [6] shows that sensory redundancy increases robustness of learning and can reinforce conveyed concepts and emotions. As described by [7], by combining auditory and visual cues and using sensory redundancy, Western culture creates audio and visual experiences that are easily recognized and embedded into a collective pool of shared cultural knowledge, and which are recognized across generations and continents.

While the concept of the cultural ubiquity of music is a prevalent thesis, it may not completely apply to people who are Deaf (D) and hard-of-hearing (HOH). These individuals tend to have more limited access to the emotional experiences, collective culture and shared knowledge that is offered for a larger hearing (H) culture through the auditory system [3]. However, this does not mean that Deaf culture is completely devoid of music; Deaf individuals experience music through vibration where the bass frequencies can be felt through the skin and air cavities of the body. In addition, sign language can be presented in a rhythmic fashion [3].

Although HOH individuals have residual hearing, the extent of it varies. Some individuals require hearing aids, whereas some do not. The issue is that hearing aids distort music by performing non-linear compression on auditory signals. Many individuals, who lose their hearing over time, report that hearing aids result in the inability to recognize important features of music, such as melody, amplitude modulation and frequency pitch [8]. Other individuals may experience loss of certain frequencies such as high or low frequencies, which results in incomplete emotional information. Consequently HOH individuals, even though they retain some hearing, are more visually oriented and are shown to require visual cues to complement speech information [9]. Hence, HOH individuals may benefit from having access to music through visuals combined with tactile.

One of the solutions to providing more access to D/HOH individuals is through a sensory substitution method called music visualization [10]. Modern music visualizations have existed for almost a century and tend to have one of three purposes; artistic expression, for learning music theory and as visual entertainment that accompanies a sound track. An example of artistic expression can be seen in Disney's *Fantasia*, which provides a visual rendering of music where various degrees of abstract and literal art were used to communicate emotional and entertainment properties of the music [3, 11]. Also, several modern artists have created abstract renditions of their own music (see [11] for examples). Such endeavors require large amounts of time and effort among production and animation teams for a one time visual interpretation, which is not feasible for every musical piece. For the purpose of this thesis, music

visualization is an automated technique that generates a visual representation of a piece of music without human intervention.

The field of automated music visualization systems has existed since the advent of modern computer systems. However, the existing automated music visualization systems are not intended to convey any type of emotional value to users, least of all to D and HOH individuals. This is because they typically tended to show visual renditions in a static form for the purpose of learning music theory (see [12], [13] and [11]) instead of communicating the emotional aspects of music.

Music visualizations that are used as visual entertainment tend to exist in relation to commercial music visualizations such as those offered in Windows Media Player™, iTunes™, PlayStation 3™ and WinAmp™. These music players use algorithms that extract rudimentary beat information from a wave file. The result is displayed using various colourful and animated visuals, which can be misleading the viewer because it is possible to have fast changing visualization for a slow and sad song potentially confusing a D/HOH audience [3]. Whereas the music visualization solutions that are designed to teach music theory suffer from being too informative, the commercial solutions suffer from the lack of enough information to enjoy the emotive nuances involved in the sound-based music. .

In this thesis, I have designed and developed a music visualization system, called MusicViz, that is combined with a vibro-tactile system called the Emoti-Chair to create a cross-modal music entertainment system called VITA (Visually Immersive and Tactile Animation). MusicViz is a unique music visualization system because its primary goal is not for education or simply to be aesthetically pleasing; its goal is to combine the entertainment aspects of music with the informative aspects such that HOH and D individuals can enjoy the emotional aspects of music. To do this, I have designed MusicViz with a translation algorithm that converts auditory signals to visual signals using a psychological model developed by [2, 14] of how pitch, amplitude and tempo are translated between the auditory and visual domains (e.g. humans perceive visual height, brightness and visual events per time to be equivalent to

pitch, amplitude and tempo in the auditory domain). Can users differentiate the three basic building blocks of music when combined into a single visual/tactile system using VITA?

As part of my research, I also wanted to assess users' emotional experiences of VITA. As such, I carried out user studies to explore the enjoyability and emotional impact of the visual and tactile music generated through VITA. With traditional music, a user can experience happiness, sadness, anger and fear as well as complex mixes of these emotions [4]. However, if users cannot hear a song very well, is it possible to experience these emotions through the use of VITA? Lastly, even though [15] reported that 94% of people surveyed found vibrating devices and visual devices useful for experiencing music, my question is: does presenting users with VITA provide them an emotional experience?

1.1 Thesis Statement

This thesis explores the concept of experiencing music for the purpose of enjoyment using visual/tactile stimuli for individuals who are HOH or D through a new music visualization called MusicViz. MusicViz is a sensory substitution algorithm that translates the musical properties of pitch, tempo and amplitude into equivalent visual stimuli based on a pre-attentive model of psychological audio and visual equivalences of pitch, amplitude and tempo. In addition, MusicViz is combined with a tactile feedback system called the Emoti-Chair to display the auditory cues through vibrations. The combination of the Emoti-Chair and MusicViz create a system referred to as VITA (Visually Immersive and Tactile Animation). VITA makes it possible for a user to experience music as visual stimuli alone or in conjunction with tactile stimuli.

The main hypothesis of this thesis is that it is possible for D or HOH users to enjoy music through the tactile and visual stimuli presented by VITA. The underlying theory informing this hypothesis is a sensory substitution model based in perceptual psychology that provides a set of visual equivalents to the fundamental musical concepts of pitch, amplitude and tempo normally expressed through sound. VITA is

able to accept a MIDI file as input and automatically, in real-time, render the song visually on MusicViz and tactically on the Emoti-Chair. The details in the algorithm are discussed in the Section 3

A user study was conducted for this thesis in order to examine the enjoyability of music through the MusicViz and Emoti-Chair combination for D and H users. The study consisted of showing six songs, one from six different musical genres to each participant and having them rate their enjoyability, emotional information on two emotional scales (continuous and discrete), their levels of focus on the visual and tactile components, and levels of agreement with statements pertaining to the aesthetics, patterns and shapes of the visualization as well as the comfort level of experiencing music through the vibro-tactile component. The songs were randomly presented through one of three playback conditions: (1) percussion to the Emoti-Chair and instruments to MusicViz; (2) percussion to MusicViz and instruments to the Emoti-Chair; and (3) percussion and instruments to both devices. The user study showed that VITA was an enjoyable method of experiencing music and participants thought that the visualization and Emoti-Chair offered an entertaining insight into music. The results showed that D/HOH participants had self-reported emotional responses to music that are similar to those of hearing participants. It was also found that there were differences in the enjoyability between D and H individuals where H pariticipants found the vibro-tactile aspect of VITA more enjoyable than their D counter-parts. It was originally hypothesized that D participants would have an easier time experiencing the visualization and vibrations compared to their H counter-parts. Reasons for this include D participants already access music through alternative methods such as vibrations by being in close proximity to speakers; however, this was not the case. In addition, it was hypothesized that Rap/Hip-Hop/R&B (RHRB) music would be the most enjoyable for D participants; however, this was also disproven.

1.2 Research Questions

The main goal of the user study was to test the effectiveness of the VITA system and receive feedback from users. The following research questions were formulated to address this goal.

1. What is the impact on the experience of users from different musical genres with VITA?
2. What are the differences in the experience of VITA between H and D audiences?
3. What is the impact on users of the different VITA conditions (e.g., separating the drums from the instruments, playing the instruments on the Emoti-Chair and the drums on the visualization or vice versa, or the playing the instruments and drums simultaneously on the Emoti-Chair and MusicViz)?

1.3 Contributions of the Thesis

There has been some research already conducted on accessing music through visualizations for educational purposes or pure aesthetics, and vibrations for accessing auditory cues from films. However, the concept of combining a music visualization system with a vibro-tactile feedback device, for experiencing the emotional aspects of music, has had limited research. The contributions of this thesis are listed below.

1. The main contribution of this thesis was the music visualization system (MusicViz). The algorithm, which performs the visualization of music is based on published psychological models of auditory perception for mapping auditory cues to visual cues. This algorithm automatically translating pitch, amplitude and tempo to visual constructs. Percussion is split from the other instruments and visualized using toroid objects. The instruments are visualized using coloured pipes, where the colours indicate similar instruments. Pitch alters the y-axis position of pipes; amplitude alters each pipe's size; and tempo is displayed by the amount of movement occurring over time. MusicViz was created for people who are D/HOH to experience music visually without needing any training and for the purpose of enjoying that music. The translation algorithm developed to implement the auditory to visual mapping which is based on how humans link frequency, amplitude and tempo from the auditory domain to the visual domain. The contribution is furthered by combining a tactile feedback device (Emoti-Chair) with MusicViz for a cross-modal system for experiencing music through visuals and vibrations (VITA) with the goal

of providing and enjoyable experience for D and HOH listening to music. The user study showed that VITA did meet its goal, which is to provide an enjoyable and informative musical experience. Individuals were able to use the vibrations to learn how the visualization works and vice versa; and individuals were able to have an enjoyable experience. The user study revealed that VITA evoked self-reported emotional responses to music, such as happiness, sadness, anger, fear, that are similar to self-reported emotional response to hearing music. In addition, the results show that the emotional responses, enjoyability, and the amount participants focused on MusicViz and the Emoti-Chair differed based on hearing status, playback conditions and genres. Even though the results show that VITA achieves its goal of providing users with emotional insight to music, more participants are required to examine all the effects of VITA.

2. Another contribution of this thesis is the user study. Music visualization systems have been created for use in education and commercial visualizations are used for entertainment purposes as described in section 2. However, there have been few user studies conducted into the effectiveness of these systems at communicating the emotional and entertainment aspects of music. The other music visualization systems, described throughout the literature in section 2, were created with a goal of teaching music or communicating the pitch, amplitude and tempo changes in music; however, these systems were not tested by the creators. Another of my contributions is that I examined the effectiveness of music to provide an enjoyable musical experience where the individuals were able to experience emotions that music either expresses. VITA is tested in formal user studies with deaf and hearing individuals. The study was the first to compare the experience of different genres identified in the literature [16], hearing conditions and playback scenarios for MusicViz and Emoti-Chair where the different playback scenarios divided percussion from other instruments. In addition, it is the first study performed into cross-model sensory substitution systems for the purpose of experiencing the enjoyable and informative aspects of music using a combination of instruments that are internally and externally validated and published. The results of the study provided insight in how H and D individuals perceive

different genres, experience emotions with and focus on the music visualizations and the vibrations produced by the Emoti-Chair alone as well as with the combination of visualization and vibration. The results showed that, of the three conditions used, having all of the music played through the Emoti-chair and MusicViz together was best.The user study results also demonstrated that participants found the combined system enjoyable, entertaining and informative. Participants' comments state that they would purchase such a device for personal access to music. This indicates that it is feasible to provide access to music for D and H individuals. It is promising that the system is taken seriously for accessing music unlike previous music visualization systems in the commercial environment.

3. The mapping algorithm for automatically translating auditory to visual cues that I created was designed to allow the users to experience the visualization with no training which is typical for those using visualizations for educational purposes. The results of the user study indicate that participants were able to experience emotions as defined in [4] when exposed to the visual and tactile stimuli. In addition, the results show that participants were able to carry out the study tasks without any training.

2 Literature Review

2.1 Introduction

This chapter presents the literature explaining the background information about the VITA system. VITA is composed of two different sensory substitution systems for the deaf and hard of hearing. The first system is MusicViz, which is a music visualization system. The second system is the Emoti-Chair, which presents auditory music as tactile vibrations. This chapter presents VITA's background information in two sections.

The visualization section provides background information pertaining to the field of visualization. This includes a brief history of visualization, its development in the 20th and 21st centuries and the rise of computer visualization systems. In addition, a summary of major commercial and academic visualization systems dating from the 1970s is presented. Following this, the disadvantages of these systems in communicating emotional information to the deaf and hard of hearing is discussed. MusicViz is presented as a solution candidate. MusicViz is differentiated from prior visualization techniques because its visualization algorithm is based on psychological research. This research is presented and the reason for its improvement over existing visualization solutions is discussed.

Lastly, the *Vibrotactile* section discusses the literature explaining the background information pertaining to human skin's ability to discriminate the complex signals of music. This includes a discussion of the physiology of the skin and the frequency theory of auditory signals. This is the background information on which the Emoti-Chair bases its auditory-to-haptic translation algorithm. The section also presents another prominent haptic feed-back device developed by the authors of [15].

2.2 Visualization

Visualization dates back to the 6th millennium BCE with the earliest known visualization being a map of the known-world in 6200 BCE [17]. Visualization existed as a means to aid in navigation, understanding star and constellation positions [17]. From the 17th century onward, advancements in analytical geometry, theories of error of measurement and the birth of probability theory increased the rate at which data was collected [17]. The 18th and 19th centuries led to socio-economical advancements, where complex economics and census information pertaining to people was collected; such information included: social information, medical information and economic statistics [17]. These advances, in conjunction with large scientific leaps in physics, mathematics and sciences, sped the increase of data collection and due to the large amounts of data, became very hard to analyze. The need to efficiently analyze large data sets acted as a catalyst for the development of visualization techniques [17].

[17] states that "the birth of statistical thinking was accompanied by a ruse in visual thinking: diagrams were used to illustrate mathematical proofs and functions. Various graphic forms were invented to make the properties of empirical numbers … more easily communicated [and] accessible to visual inspection" [17]. He suggests that due to the explosive growth in data collection, visualization in the 19th century expanded. Modern forms of visual statistics were developed to aid finding trends in large data sets. Such visual forms included bar charts, pie charts, histograms, line-graphs, time-series plots and contour plots [17].

Due to the large amounts of research into computer technology during the Second World War to break encryption codes and create advanced wireless communication systems, large amounts of research into digital computers began in the 1950s. Digital computers provide more advanced display and input technology for the end-user, which provide new paradigms for visualization. Graphical user interfaces aid in the creation of human-like and intuitive interactions with machines. The field of human computer interaction is concerned with the development of efficient computer interfaces. HCI includes other fields such as psychology, art and health science for research into the ease of use of computer system and strain they put on the user.

Kosara (2007) states visualization is a broad term with no clear definition [18]. The term differs between fields such as computer science, engineering and illustration. Examples of visualization from Kosara (2007) are: "architectural visualization, terrain visualization, 3D medical/volume visualization, 2D or 3D flow visualization, flow topology visualization, information dashboards, music visualization, photomontage or collage, traffic signs, traffic signals, sign language, icons, visualizing oneself in a different job/situation, visualization of concepts, visualization of [abstract] concepts, drawing fractals, etc.". Even though there is no clear definition, visualization is generally characterized by being based on data and producing an image that is readable, recognizable and informative for a specific task. He rates visualizations on a 1D sublime linear scale ranging from pragmatic to artistic visualization (see Figure 1). Sublime is described as artistic visualizations "that … inspires awe, grandeur and evokes deep emotional

and/or intellectual response" [18]. The opposite, anti-sublime (pragmatic), is used in computer science and validated through user studies to remove subjectivity in order to facilitate in easy understanding. Kosara's paper describes music visualization as purely sublime due to lacking recognizability and readability.

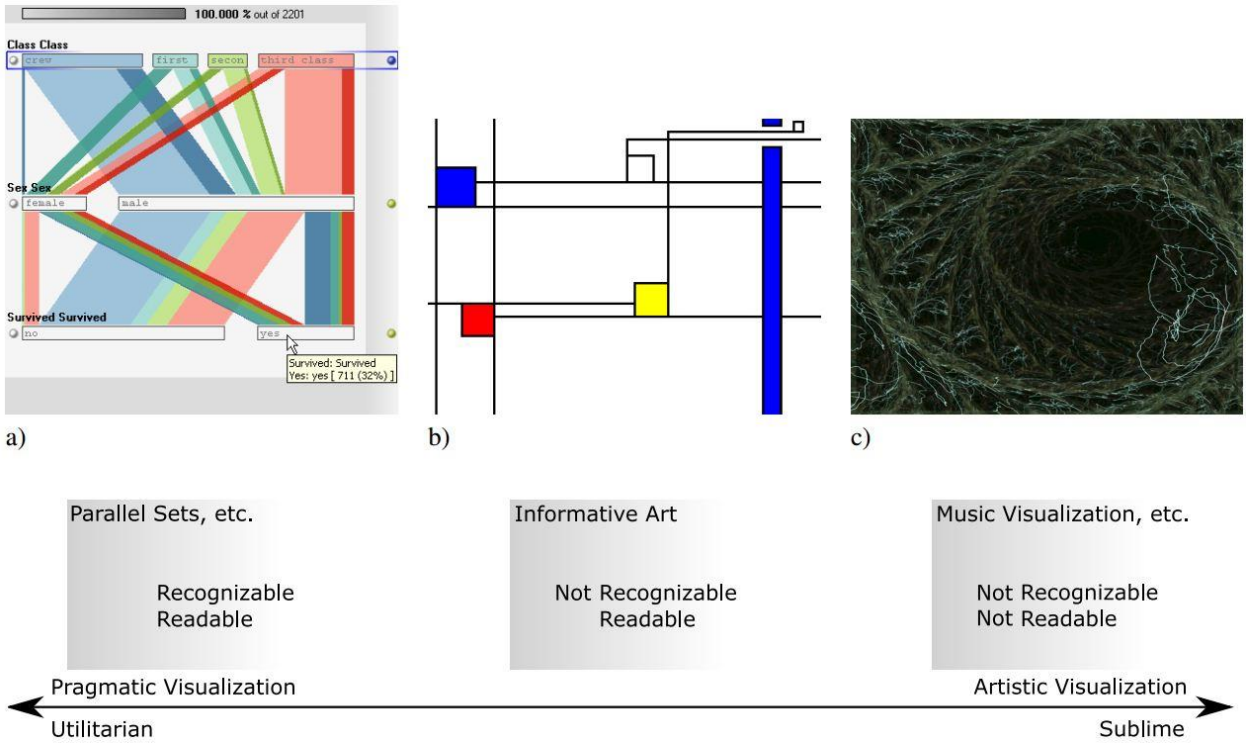


Figure 1: "The gamut of data-based visualization. a) Parallel Sets … show data about the people on the Titanic, and are readable and recognizable as a visualization; b) Ambient visualization … visualizing a bus schedule are readable but require more effort and are not readily recognizable as a visualization; c) Music visualization like MilkDrop … is also based on data, but not readable" [18]

Pragmatic visualization is further classified into two groups by [17]: information visualization and scientific visualization. Information visualization is concerned with producing visual presentation of data sets that lack "inherent 2D or 3D semantics and therefore also lack a standard mapping of the abstract data onto the physical screen space" [19]. Abstract data is referred to as data without a spatial mapping to geometry, whereas scientific visualization is concerned with existing 3D structures [17, 20].

The various definitions of visualization are relevant to this thesis due to the nature of music visualization. Considering music visualization is the visual rendition of music and music is composed of frequencies, amplitudes and timing; it is technically numeric information. However, music is more than a collection of notes and volumes. The way in which the notes and volumes are composed, define the timbre [2]. It is through timbre that "[music] is capable of arousing deep and significant emotion" [21] in individuals who experience it [21]. The interesting aspect of music is that even though it is composed of numerical information, individuals do not listen to music for its informative aspects. Individuals listen to music for its power to evoke and represent emotions [4]. Sloboda (1992) reports that "music relaxes [individuals] when [they are] tense and anxious" [21], "[individuals feel] understood and comforted in pain, sorrow and bewilderment" [21], "involvement in music [detaches individuals] from emotional preoccupations" [21], others are "inspired to be better [people]" [21] and music shows that "emotions are shared and not your burden alone" [21]. In addition, the creation of music is considered an artistic process for expressing emotions and feelings, not an informative/mechanical process. Even though music is composed of sound information, both the creation and consumption of music is characterized by emotional experiences. Emotional experiences are transmitted through the changes in timing, amplitude and frequency.

Music visualization is either purely a sublime visual rendition, with little recognizability and zero readability; or too informative. It does not contain a mix of sublime and pragmatic aspects. By missing the informative aspects, music visualization lacks the data, which transmits emotional information. This thesis focuses on creating an informative (pragmatic) yet entertaining (sublime) music visualization system for the deaf, deafened and hard of hearing. In order to have a pragmatic and sublime visualization, the research focuses on incorporating psychologically justified translation principals for translating auditory cues to visual cues. The music visualization system created for this research is referred to as MusicViz. MusicViz is interfaced with the Emoti-Chair, which is a haptic feedback chair that presents music as tactile vibrations along the user's back. The combined MusicViz and Emoti-Chair system is referred to as VITA (Visually Immersive and Tactile Animation).

2.2.1 Visualization as Sensory Substitution

Music visualization is an example of sensory substitution, where as traditional forms of visualization are not. Sensory substitution is strictly the translation of data from one human sensory system to another human sensory system. Example of such systems are visual-to-tactile, tactile-to-tactile (prosthetics), audio-to-tactile, audio-to-visual [22]. Visualization of scientific data is not considered a sensory substitution technique because it does not take information from one human sensory system and translate it to another sensory system. Music visualization is an example of human sensory substitution because it specifically translates/interprets auditory information to visual information. The *Music Visualization* section discusses pragmatic, sublime and hybrid music visualizations, which are forms of sensory substitution because audio information is converted to visual information. Each section contains descriptions of other researchers' techniques, such as their algorithms and systems. These systems and algorithms constitute the background literature into audio-to-visual sensory substitution techniques for music.

2.2.2 Music Visualization

Music visualization exists in three forms in research: pragmatic visualization, sublime visualization and hybrid (sublime/pragmatic) visualization. Pragmatic and sublime visualization are differentiated by the underlying theory that translates auditory cues to visual cues. Pragmatic visualizations attempt to convey music theory in simpler notation conducive for learning music theory. They do not attempt to create visually appealing visualizations. These visualizations tend to be just as complicated to an untrained musician as formal notation; however, they may provide more information and insight to composers and other musicians. Other pragmatic visualizations, which target a general audience, tend to be for utilitarian purposes such as genre categorization for unknown songs or visual sorting tools for large music collections.

Unlike pragmatic visualizations that attempt to teach music theory, sublime visualizations do not. Sublime visualizations tend to extract rudimentary beat information to produce pulsing lights to animate a

seemingly random set of moving abstract shapes and aesthetic colours [3]. These visualizations provide pretty picture but do not provide insight to the music [3].

Hybrid (pragmatic/sublime) visualizations attempt to compromise information and entertainment to achieve equilibrium. Hybrid visualizations attempt to convey informational aspects to the user as well as entertainment aspects. The information in hybrid visualizations is geared towards diluting complex musical notation through aesthetics to target the general population. The final goal is to provide a system that conveys musical information, yet is aesthetically pleasing.

MusicViz is a hybrid music visualization system created for this thesis. MusicViz resulted from the background literature into pragmatic, sublime and other hybrid visualization systems. It attempts to improve on the disadvantages of the various reviewed systems by providing a visualization algorithm grounded in psychological theory. Its goal is to provide music visualizations to individuals who are D or HOH for the purpose of experiencing emotions from music.

2.2.2.1 Pragmatic Music Visualizations

In terms of pragmatic visualizations, numerous exist. [12] presents a static music visualization called Structured Event Editor (SEE). SEE allows for the construction of custom musical predicates to control the mapping of musical constructs to the visual medium. Music predicates allow for the user to design a custom style file, which dictates the final visual's parameters. The SEE program constructs data structures for the visualization by translating the audio constructs from a data file containing musical information as sheet music. It is unclear, whether MIDI is the source data.

SEE produces multiple orthogonal and perspective views of the musical data. The orthogonal view is a colorless 2D representation of music in a piano-roll fashion with black dots. The orthogonal view can be created from any angle of the 3D visualization. In the figure below, the music is visualized based on time and pitch. Time is mapped onto the x-axis and pitch is mapped to the y-axis [12].

Figure 2: Static orthogonal view of visual music [12]

In addition to the orthogonal view, SEE contains a 3D perspective view [12]. The perspective view allows the user to view the music in a 3D environment. Like the 2D perspective, the x-axis represents time and the y-axis represents the frequency [12]. The notes are represented by coloured cubes and the colour is based on the pitch [12]. The system allows for the user to view the musical piece through different orthogonal and perspective viewpoints based on the supplied style sheet [12]. An example of a 3D visualization is shown below.

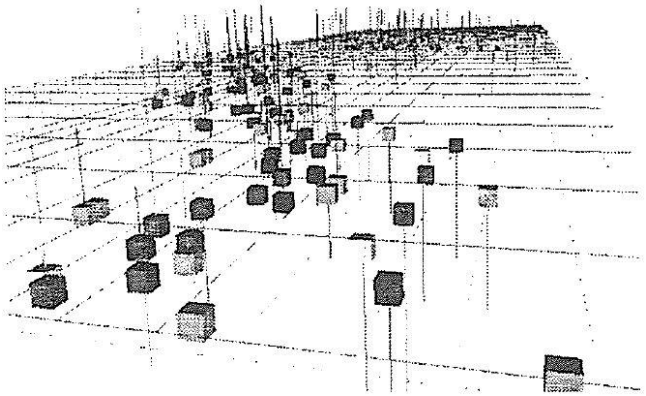


Figure 3: Static 3D view of the music composition [12]

The authors of SEE [12] offer an interesting way to visualize music; however, the specific application of the visualization is not clarified. The authors state that the program incorporates a model allowing for the visuals to change overtime in order to facilitate time-dependent interpretation. However, it is impossible to incorporate time-dependent interpretations without animation. Even though the use of SEE is not

stated, it is clear the visualization is not geared towards the hard of hearing and deaf due to three reasons: (1) the system lacks to deliver any amplitude or rhythm information to the end user; (2) the system does not animate the visualization; and (3) the visualization is not grounded in any psychological research based on sensory substitution. It is assumed, that presenting a static music visualization based on pitch and time, the researchers' goal is to assist in musical education through the use of less complicated notation. However, if this assumption is correct, the effectiveness of this type of visualization is further compromised due to the lack of user studies into the system's effectiveness.

The MISUAL system [23] is another example of a pragmatic visualization. Like the previous research presented by [12], MISUAL provides an objective and static information visualization "regarding musical [impressions] that … [enable] people to efficiently categorize a huge number of musical pieces [through visuals]" [23]. MISUAL translates auditory wave form music by visualizing the elements of music as geometric shapes and colours [23]. MISUAL's goal is to allow an individual to visualize their impression of music, such that it allows a user to visually categorize large music collections [23].

MISUAL performs category visualization in two steps [23]. The first step is volume transition and the second step is repetition identification [23]. Volume transitions create a cylindrical shape communicating whether a musical piece contains vocals and the extent at which the vocals are present [23]. The volume transitions are calculated using power information, which are smoothed using a moving average [23]. The authors do not define power information; however, it is assumed to be the song's volume at a point in time. It is unclear how utilizing volume pre-attentively communicates genre information. For example, classical and rock pieces can be loud; however, it does not mean both genres are related. The second step is repetition identification [23]. A musical piece contains multiple sets of repetitions, which are coloured differently. Repetitions are calculated using the Mel-frequency Cepstrum Coefficient. The final visualization is shown in Figure 4.

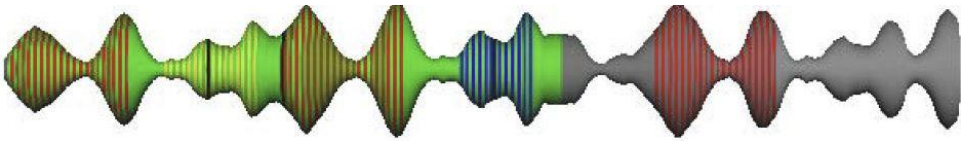


Figure 4: A final static visualization of music from MISUAL [23]

As previously stated, MISUAL's goal is to provide an intuitive static visualization of the music; however, there were no user studies to show the intuitiveness of the visualization. The final visualization is reminiscent of a sinusoidal wave used in an oscilloscope, which conveys little to no emotional experiences. The visualization in Figure 4 is a representation of *Eine Kleine Nacht Musik* by Mozart; however, ascertaining information or emotional experience from this visualization without any training is not readily obvious. For instance, a song with many vocals and long durations of repetitions is a general description of music, which can be a description of death-metal; however, Figure 4 is not a visualization of a death-metal song. It is difficult to judge musical genre based on amount of vocals and repetitions because many genres contain vocals. As an example, rock, pop, opera and country contain vocals. In addition, there are many different sub-genres of rock such as classical rock, industrial rock, punk rock, etc. Different countries and cultures also create personalized spins on these genres. Industrial rock from Germany sounds different than industrial rock from Canada and different than industrial rock from United States. Considering it is difficult to pre-attentively differentiate classical music from rock music with the MISUAL system, how can an individual expect to classify large music libraries when the differences between genres are so fine?

[24] presents further research into static pragmatic visualizations for genre categorization. This paper covers multiple different techniques for music information retrieval visualization. The commonalities between all the techniques in [24], including MISUAL's, are audio parameterization [24]. Audio is converted to data using Mel-Frequency Cepstral Coefficients, Fourier transforms and/or component analysis [24]. The resulting datasets are reduced in dimensionality and mapped to geometric data [24].

Once data is parameterized and its dimensionality reduced, music data is visualized alone or in relation to other musical pieces for categorization [24].

Many visualization techniques are described for visualizing a single piece of music. One method is the use of a similarity matrix (Figure 5 and Figure 6), which is a "technique for studying the global structure of time-ordered media streams" [24] that allows for the user to see the similarity of two points in a song [24]. The similarity matrix differentiates the chunks of music such as the intro, verse, chorus and bridge [24] sections.

The similarity matrix provides insight to the beat periodicity and beat strength in a song, which is visualized using beat spectrograms and beat histograms [24]. The authors describe beat spectrum (Figure 7) as "a measure of self-similarity as a function of the lag" [24], where peaks in the spectrum correspond to the amount of audio repetition at that specific rate [24]. The goal of a beat spectrum allows for the analysis of rhythmic variations over the life of the song [24]. A beat histogram (Figure 8) is similar in that it visualizes beats; however, "it visualizes the distribution of various beat-level periodicities of the input signal" [24].

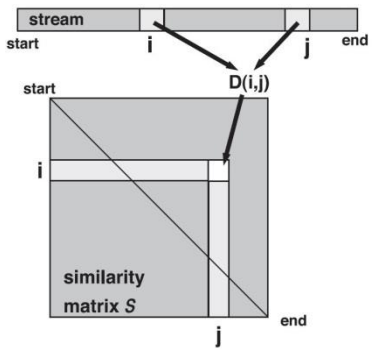


Figure 5: Creation of a similarity matrix from a music stream [24]

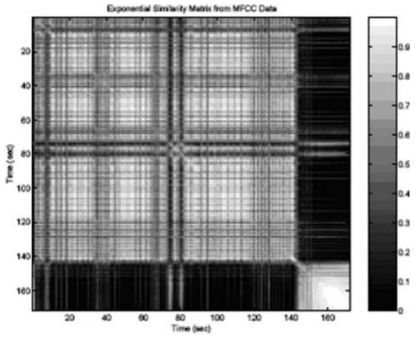


Figure 6: A similarity matrix of music for music categorization [24]

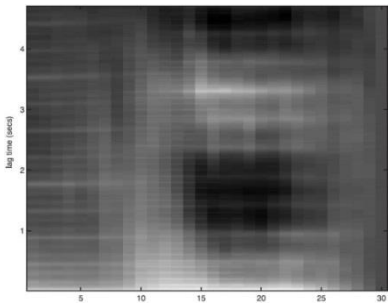


Figure 7: Beat spectrogram of a musical piece [24]

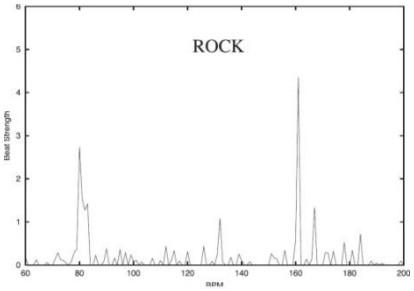


Figure 8: Beat histogram of a musical piece [24]

Timbregrams are another method to assist in pragmatic music visualization. Timbregrams are visualizations of the timbre of music, where timbre is represented by a series of vertically coloured stripes [24]. Timbregrams can be utilized with spectrograms or rectangular boxes to visualize timbre information at a point in time (see Figure 9 and Figure 10). In addition timbregrams can be mapped to 3D boxes and placed in 3D space (see Figure 11), where a box represents a song. Groups of closely spaced boxes represent musical pieces with similar timbre.

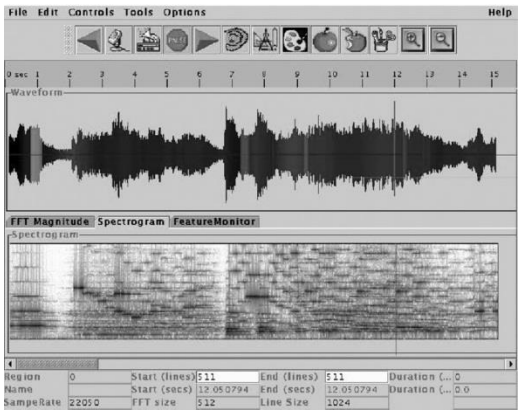


Figure 9: Timbregram using spectrograph [24]

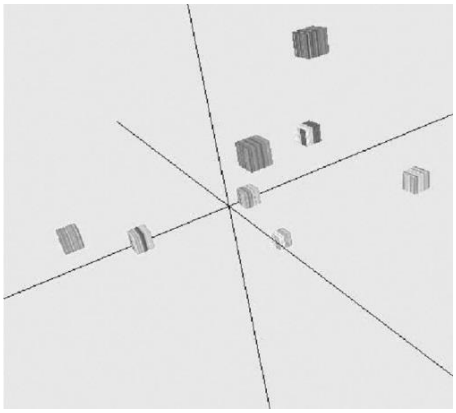


Figure 10: Timbregram using striped boxes [24]

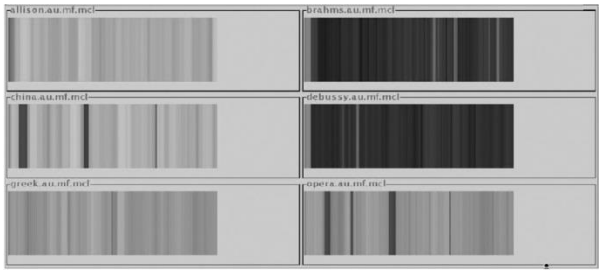


Figure 11: Timbregram of individual songs [24]

[24] presents a real-time music genre classification called GenreGram. This is the only animated visualization technique presented in [24]. The visualization program is designed to take any wave-form signal and classify the audio using fuzzy logic. The output is a confidence measure of the genres that the audio could represent. Using Figure 12 as an example, the program rates the analyzed audio signal as hip-hop with male vocals; however, it may be a male sport announcer.

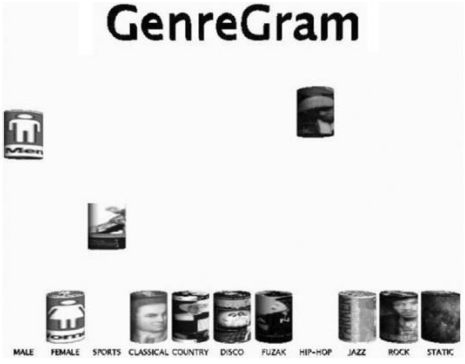


Figure 12: Real-time visual categorization of music [24]

Other categorization techniques include using self-organizing maps for visualizing genres. Similar pieces of music are grouped together on a virtual map. As the map is populated, mountains, hills and islands rise from the virtual oceans, where each geographic construct has a unique colour [24]. The more songs grouped together lead to larger geological shapes being visualized. *Islands of Music* is an example of this style of visualization [24]. It is described as a self-organizing map based on a neural network algorithm that iteratively arranges pieces of music on a grid [24]. [25] states that neural networks allow for problems to be solved where the relations between the data are unclear. The *Islands of Music* visualization utilizes a similarity measure for input to the neural network; however, it is difficult for computers to perform similarity ratings as well as humans [24]. This is due to the subjective nature of music. Musical genres contain countless sub-genres, where sub-genres can overlap and a human listen may not always classify a song into the same category as their colleague. Therefore *Islands of Music* utilizes a mix of human and computer similarity measures to personalize the visualization [24]. Figure 13 shows an isometric perspective of the *Islands of Music* visualization.

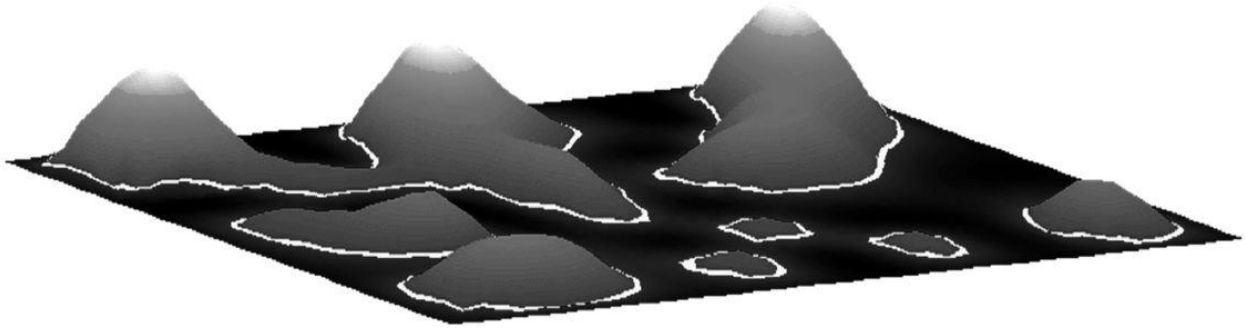


Figure 13: Islands of Music Visualization [24]

The music visualizations presented in [24] are for information retrieval and display of musical information. The issue with the visualizations is that they are not validated. No user studies are performed to test the usability of these music visualization systems, therefore it cannot be known if they are of any use to individuals.

Unlike the SEE editor presented in [12], the authors of [13] define a clear goal for their research. Their goal is to create a pragmatic visualization solution that simplifies the "complex nature of music notation" by mapping MIDI music data to 3D space using OpenGL. The MIDI data is mapped to the 3D-world based on a two-step mapping process. The first step encompasses mapping tone data. The visualization utilizes geometric spheres for displaying visual music. The physical parameter of each sphere is influenced by three bytes that describe MIDI music: pitch, volume and instrument similarity. The authors place the sphere on the vertical y-axis representing pitch . The volume value is mapped to the sphere's radius, where a larger sphere represents a louder volume. Instrument similarity is referred to as timbre. MIDI represents timbre by grouping similar instruments into MIDI channels, where each MIDI channel is coloured differently. Volume is mapped to the sphere's radius, where a larger sphere represents a louder volume [13]. The second step consists of equally spacing the MIDI channels along the x-axis and the flow of time is represented using the z-axis. During visualization, a sphere is placed on the z-axis when a note is activated. As the note is held, the sphere is elongated until the sound ceases. Shorter notes appear more spherical while longer notes are more elliptical.

Even though the goal of this research is to simplify complex musical notation, the authors did not test the successfulness of their visualization system. In addition, the authors did not base their audio-to-visual translation algorithm on psychological research; hence, the overall effectiveness of creating simpler music notation is unknown.

Hiraga, Watanabe and Fujishiro (2002) present another pragmatic visualization geared towards aiding users' understanding of musical notation by utilizing expressive cues alongside sheet music. The visualization targets users "who want to improve their understanding of a musical piece and their performance of it" [11]; however, the system is meant for use by advanced musicians [11].

Professional performers often work together to perform a piece of music. Rock bands, orchestras and jazz bands have to coordinate each other's part to create a musical piece. This includes understanding each

other's "expressive intentions" [11], which conventional sheet music does not convey. Often it is difficult because of busy schedules to meet to practice [11]. The authors present a system that allows the sharing of the musical score and emotional performance. Users are able to comment and create personalized faces to simulate the creative component of the performance.

The system proposed in this paper is a combination of the BRASS (Browsing and Administration of Sound Sources) system with previous research into Chernoff faces [11]. The BRASS system presents a focus-plus-context view of the musical score [11]. Users of the system can focus on specific parts of the composition, while maintaining its context in relation to the entire score [11]. The authors state that "based on [the user's] understanding of the performance with the score, users can place comments on a visualized figure and exchange them with other users". The goal of this is for users to "share and deepen their interpretations of a musical piece" [11].

The BRASS system visualizes the score by utilizing a spatial substrate [11]. The authors describe a substrate as a framework for utilizing spatial information to represent musical information. This is essentially a complicated title for the method of substituting music notation for simpler notation. For example, musical staffs are represented by line widths; symbols for rhythm and tonality are shown at changing points; and the number of notes in a measure is defined through brightness [11]. An example of the final BRASS visualization using Chernoff faces is shown in Figure 14.

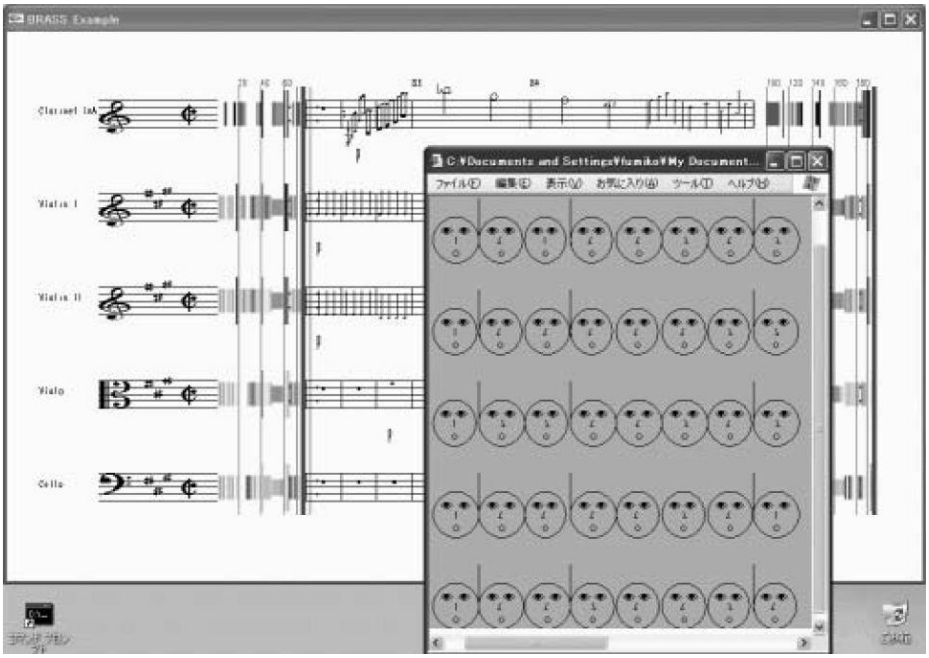


Figure 14: BRASS visualization using Chernoff Faces [11]

The research performed by [11] is unique because the researchers attempted to validate the visualization tool. However, it falls short due to only incorporating three participants, which is not enough for formal statistics.

Hiraga also presents research alongside Noriyki in the paper titled, *Visualization of Music Performance as an Aid to Listener's Comprehension* [26]. The goal of the research is to provide a static and pragmatic music visualization to aid listeners in understanding a musical performance. The authors utilize three major elements of music: (1) *tempo change*; (2) *articulation;* and (2) *dynamics change*. They are utilized to create a static visualization that expresses the emotional characteristics of music [26]. It is interesting that the authors claim to create a visualization that aids listener's comprehension, yet claim to express the complex emotions of music without utilizing colour, animation or a psychological model of audio-to-visual translations. Even though the authors claim that *tempo change*, *articulation* and *dynamics* are carriers of emotions in music, the authors do not realize that there is little overlap between the auditory and visual systems [27], and this is apparent with their study. Besides lacking to mention the number of participants, the authors excluded anyone without five years of piano experience. This means that the

system is essentially useless to non-musically trained individuals. An example of the visualization is shown in Figure 15.

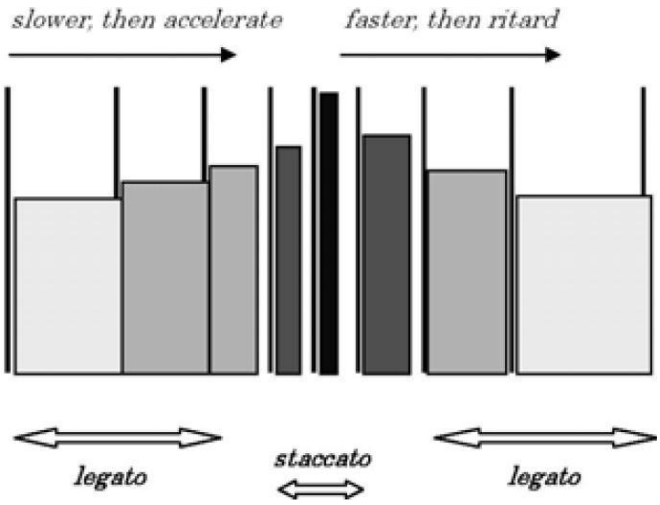


Figure 15: Performance visualization [26]

[28] presents a score-based analysis system that utilizes a visual programming system called PWGL (PatchWork Graphics Language) [29]. It is used "for computer-aided composition, music analysis and sound synthesis" [29]. The system works similarly to Max/MSP and follows the direct manipulation paradigm [29], where the user visually alters a composition.

PWGL utilizes melodic and harmonic rules for various visualizations that aid musical understanding. Melodic rules are defined to add textural information to a musical piece through Expressive Notation Package (ENP) expressions. ENP expression allows for the user to defined musical constructs such as staccatos, slurs and accents to be used; however, ENP allows for a user defined construct to replace existing notation. An example of using ENP for melodic rule visualization is shown in Figure 16. In addition, harmonic rules can be defined to visualize symmetric harmonies, which are shown in Figure 17.

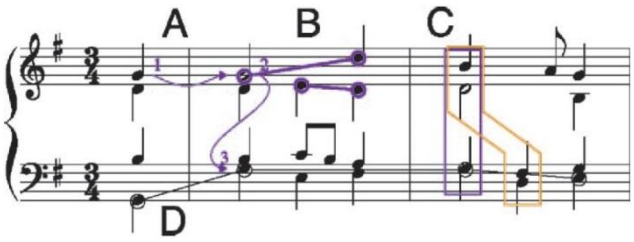


Figure 16: ENP visualization output [28]

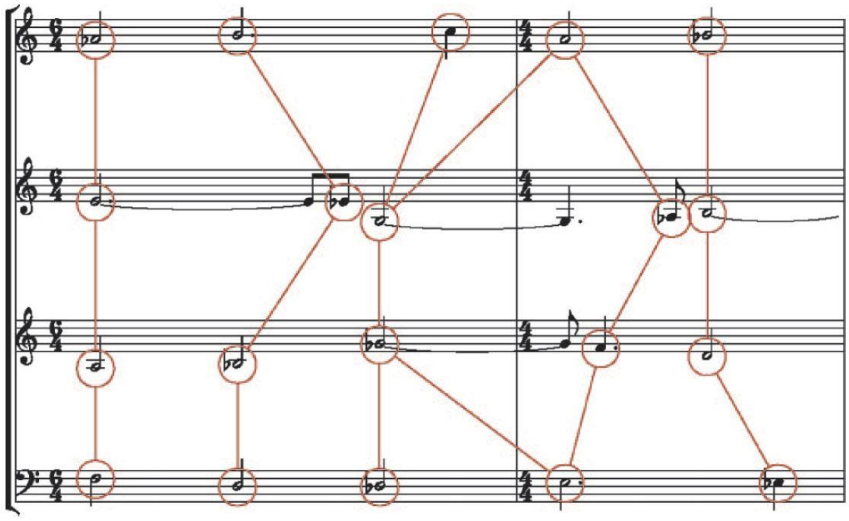


Figure 17: Visualization of symmetric harmonies [28]

The researchers state these visualizations target composers and music analysts. It is implied to not target individuals with little musical training. This is apparent when viewing Figure 16 and Figure 17; these visualizations do not attempt to create pre-attentive, informative and entertaining music visualizations. The visualizations do not substitute complex musical notation with simpler notation; in fact, the visualizations are further complicated by overlaying non-intuitive connections between notes to describe high-order constructs. A problem with the research is the authors do not test their visualizations with user studies.

2.2.2.2 Sublime Music Visualizations

The previous section reviewed pragmatic visualizations, which is conducive for learning music theory. They focus on a solution that teaches musical constructs and communicates the information of music, not

the emotional aspects. These visualizations are not widely commercially available and are never used in end-user gaming consoles such as the PS3, Xbox 360 and Wii or PC programs such as iTunes, Windows Media Player or WinAmp. The general population does not have access to these visualization systems. In addition, the pragmatic visualizations reviewed require extensive knowledge of music theory to understand the visualized information and some require manual intervention for visualization. The research in this thesis differs from the pragmatic visualizations in that my research focuses on a completely automated technique requiring no human intervention. MusicViz automatically generates an animated visual representation of music with no pre-processing.

This section reviews sublime music visualization techniques. As stated by Kosara (2007) [18], sublime visualizations "[inspire] awe, grandeur and [evoke] deep emotional … responses". Sublime music visualizations attempt to communicate the emotions of music. These visualizations appear within the commercial domain and are readily available on most computers and gaming consoles. The visualization techniques in this section present an animated emotional-centric approach through the use of animation and colour; however, results of a user study suggest they offer little emotional experience [3].

Commercial visualizations were originally introduced in the in the 1970s with the invention of the *Atari Video Music* patented in 1976 for home-use [30]. The device allows a wave-form audio source to be visualized with colours and shapes (see Figure 25). There exists four predefined shapes that can be horizontally and vertically scaled and colour nobs allow for the user to create "psychedelic displays on [their] television" [31]. A similar system is proposed by [32], which presents a software algorithm written in FORTRAN to create visual translations in real-time. [32] utilizes spectral colours to represent 24 pitches in a two octave range to convey musical emotions to the user (see Figure 26). Unfortunately the Atari and FORTRAN visualizations are not user tested.

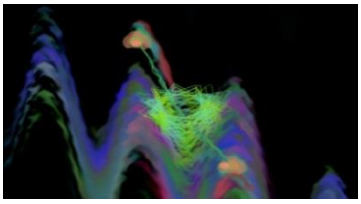


Figure 18: Windows Media Player

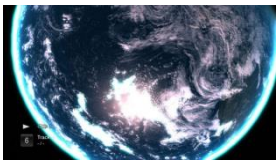


Figure 19: PS3 Gaia visualization [33]

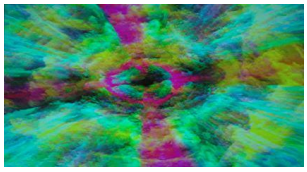


Figure 20: xBox Visualizer [34]

Figure 21: iTunes Magnetosphere

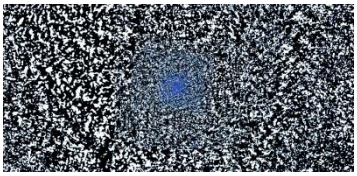


Figure 22: WinAmp

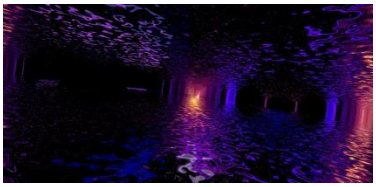


Figure 23: WinAmp

Figure 24: WinAmp

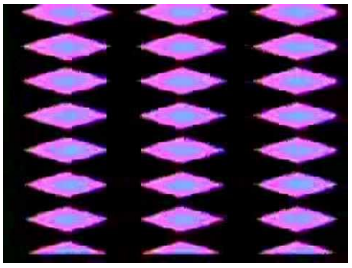


Figure 25: Atari Video Music [35]

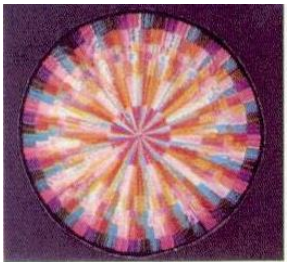


Figure 26: Painted Music [32]

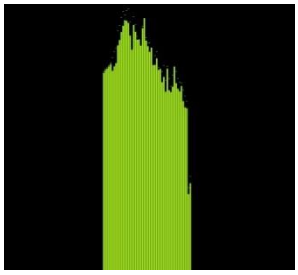


Figure 27: Windows Media Player Equalizer

The Atari style of visualization exists today as software visualizations in Windows Media Player (Figure 18, Figure 27) [36], iTunes (Figure 21) [37], WinAmp (Figure 22, Figure 23, Figure 24) [38], PS3 (Figure 19) [39], and Xbox 360 (Figure 20) [40]. However, they are any more graphically advanced than early 1970s and 80s visualizations. They are equally useless in conveying musical emotion and information[3]

and are difficult to differentiate from each other without labels. PS3's Gaia visualization is unique because it is graphically advanced. The Gaia visualization changes the camera's position during music playback to display different perspectives of Earth with different lighting conditions from the sun. Visualizations from Windows Media Player, iTunes and Xbox are hardly visually and technically impressive even when compared to their 1976 Atari ancestor; however, the Gaia visualization utilizes the PS3's advanced CPU Cell architecture and GPU capabilities to provide a hyper-realistic 3D model of the Earth [33]. The model accurately visualizes the earth's atmosphere and properly highlights the earth's oceans and light reflections from the sun [33], where "all texture data is sourced from the NASA Blue Marble Project" [41]. The Blue Marble Project contains data patched together from hundreds of high-resolution satellite images [41].

The visualization by WinAmp, Windows Media Player and Xbox result in similar visualizations to the ones created by Atari and [32]. In fact, it is difficult to identify which program made the visualizations without providing labeled figures. These visualizations utilize algorithms that extract rudimentary beat information from wave-form music [3]. The result is displayed using various exciting. There is no research pertaining to the usefulness of these systems; however, Fourney and Fels [3] present a formal user study into the effectiveness of Apple's sublime Magnetosphere. Magnetosphere is based on the idea of magnetism. It visualizes music through the interaction of charged particles. Higher amplitudes create a busier visualization with more particles. Difference in frequencies is visualized by the size of the particles. The study results from [3] suggest that Magnetosphere misleads the viewer to believe a slow song is exciting and provides one of the least meaningful representations of music [3]. Participants commented that "they [can] not really connect with the visualization … and [that] the music [is] not conveyed well" [3]. These results are not surprising because the Magnetosphere visualization is not based on musical concepts. As previously described, PS3's Gaia visualizes music as a camera moving around the earth. Gaia was originally developed as the PS3's loading screen but never used. It only appeared in an OS upgrade as a music visualization after the console's release. Considering Gaia is created as a

loading screen and its development is to flaunt PS3's graphics capabilities by creating a realistic earth, it is likely this visualization is as meaningless as Magnetosphere.

[3, 10] state that music is not always experienced through sound alone. Facial expressions and body language are expressed through dance and special effects are matched to music in movies, television shows, concerts and music videos to reinforce auditory information [3]. Auditory content is grouped with visual content since ancient times [1]. The Ancient Chinese and Ancient Greeks integrated visual expression to music over 2000 years ago [1]. However, the first recorded music videos appeared in the early 1950s and modern music videos require tens of thousands of dollars for professional film crews with hi-tech sound and video equipment [42]. [42] presents a music visualization system that automatically generates music video-like videos. This occurs by observing repetitive visual patterns in personal home videos and matching them to repetitive aural patterns [42]. The system analyzes the temporal structure and repetitive structure of the video and audio [42]. The algorithm defines a set of repetitive segments to match to the audio's prelude, interlude and coda [42]. The authors provide results of a user study with 10 participants. Each participant is shown 15 music videos from two other automated music video generation methods. Each participant assigns a satisfaction rating to the video. The authors state that their automated music video generation system has the highest score; however, there are no statistical methods applied to the data besides generic averages. It is impossible to know if their system provides significantly different results in enjoyability.

2.2.2.3 Hybrid Music Visualizations

[3] present an analysis of the *Music Animation Machine* (MAM) [43]. MAM is a visualization tool that provides various real-time renditions of music based on MIDI data. It contains a plethora of visualizations such as the Piano Roll, Circle of Fifths Colour Wheel, Interval Types, Harmonic Staffs, Tonal Compass, Part-Motion, etc. All MAM visualizations are shown below from Figure 28 to Figure 39. The research presented within [3] is unique because proper user studies are conducted pertaining to the effectiveness of the music visualizations in conveying emotions to the hard of hearing and deaf. Even though MAM

provides numerous visualizations, [3] conducts user studies into three of the twelve visualizations: the piano roll; part motion and tonal compass.

The piano roll displays musical information using colored bars. The bar's y-axis position represents the note's pitch and the x-axis position denotes the note's timing relative to other notes within the composition [44]. The bar's color denotes the note's timbre such as the instrument family [44]. The part motion represents notes as transparent balls [44]. Each ball size represents the length of the note and a line segment connects sequential notes [44]. During animation, the ball animates from its start position and decreases size until the new note is triggered [44]. The tonal compass arranges notes based the circle-of-fifths model [3]. Each pitch creates a circle of a certain size and positions the circle a certain distance from the center of the visualization. The animation is a spinning circle that attempts to "move toward tonal equilibrium (the point at which its center of gravity is lowest)" [44].

The piano roll is described as "essentially meaningless without some explanation of what [is] happening" [3] and the tonal compass is equally uninteresting and meaningless to the audience. However, three participants find the balls animation the most useful of the three visualizations studied. This leads the authors to state that "while users want to be entertained, simple representations that communicate a sense of what is happening in the music might provide useful … [when] combined with entertaining views" [3].

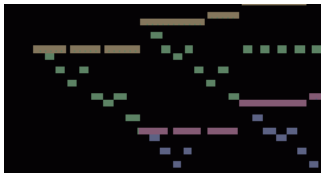


Figure 28: Piano Roll [45]

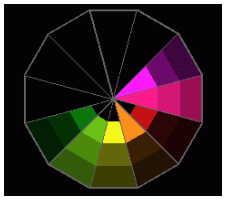


Figure 29: Circle of Fifths [45]

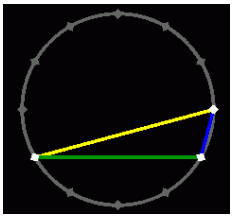


Figure 30: Interval Types [45]

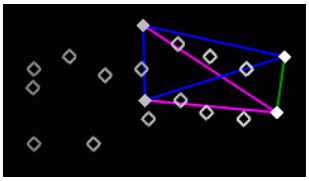


Figure 31: Interval Types [45]

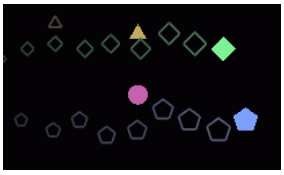


Figure 32: Shapes [45]

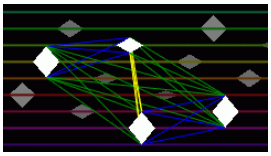


Figure 33: Harmonic Staff [45]

Figure 34: Harmonic Compass [45]

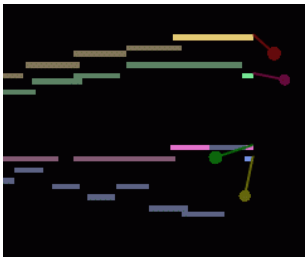


Figure 35: Piano Roll Harmonic Compass [45]

Figure 36: Triads [45]

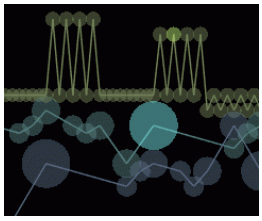


Figure 37: Part Sequence [45]

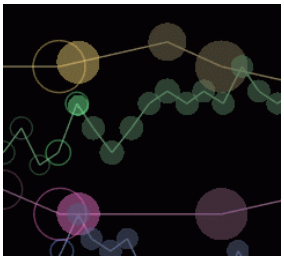


Figure 38: Part Motion [45]

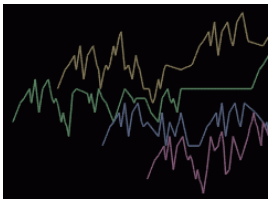


Figure 39: Part Trajectory [45]

In addition to presenting a user study analyzing MAM's effectiveness of conveying emotion and information to the hard of hearing and deaf, [3] also present a user study analyzing the effectiveness of the Motion Pixels of Music (MPM) visualizer (see Figure 40). The Motion Pixels animation is a visualization system created at the IMDC lab. It is a precursor to the MusicViz system. The MPM system

displays instruments as fans and each object contains a picture of the instrument it represents. When an instrument is played, a fan-blade is animated as an outward or inward decay [3]. Drums are displayed as circles that pulse. MPM is a real-time music animation system that uses MIDI input and attempts to create a visually appealing yet informative visualization to convey music. Participants consider MPM as one of the more informative visualizations because it best indicates how the song could sound. However, the participants indicate the visualization is boring and too busy to allow for entertainment.

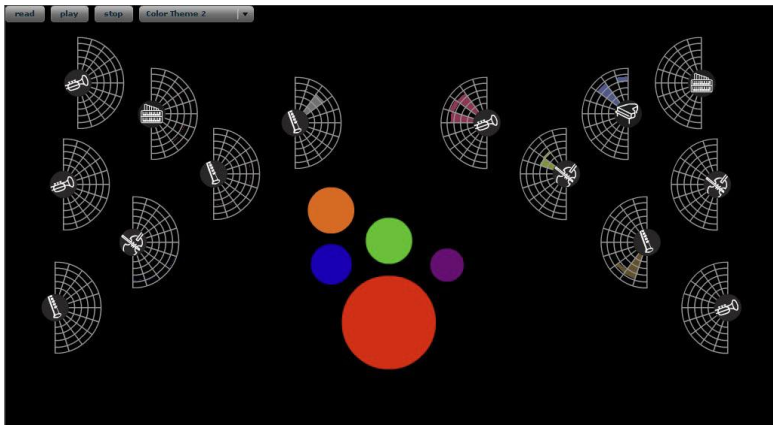


Figure 40: Motion Pixels of Music [3]

[46] presents a unique visualization solution in that it is a hybrid of sublime and pragmatic visualizations. The goal of this paper is to provide a visualization that is not only pleasing but also visualizes the relationship between complex harmonic relationships for musically untrained individuals [46]. The authors "put emphasis on modeling harmony: affinity of tones and consonance with continuous colors" [46]. The authors state that existing music visualization systems, that utilize color for tones, visualize pitch classes, major and minor chords; however, they do not visualize tone combinations. [46] proposes accounting for volume and change in volume overtime in conjunction with dyads [46]. Dyads are two are more tones in music that are played concurrently. Their method utilizes a key spanning circle of thirds that is mapped to a color wheel. Colours are represented as vectors [46]. When multiple tones are triggered, a vector is created for each tone. The tone volume represents the length of the vector. The

vectors are added together and the result determines the hue and the vector's length represents the saturation [46].

The purpose of the visualization is to display consonant dyads (two or more pitches) as saturated colours [46], where dissonant tones have low saturation and consonant tones have high saturation [46]. An example of their system is shown in Figure 41. The figure is an example of Pachelbel's Canon in D major. Considering the piece is played in D major, the corresponding visualization's colours are situated around orange as assigned from the colour wheel (see Figure 42) [46]. The visualization can be seen to alternate from A major to D major to B flat major to D major and back to G major [46].

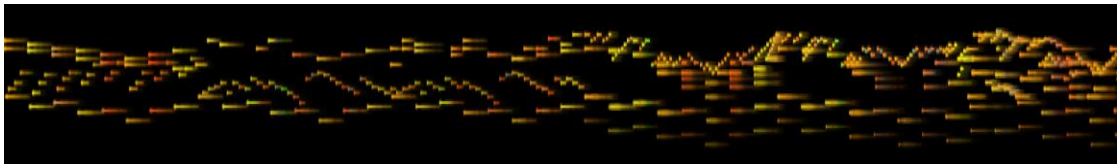


Figure 41: Dyad Visualization [46]

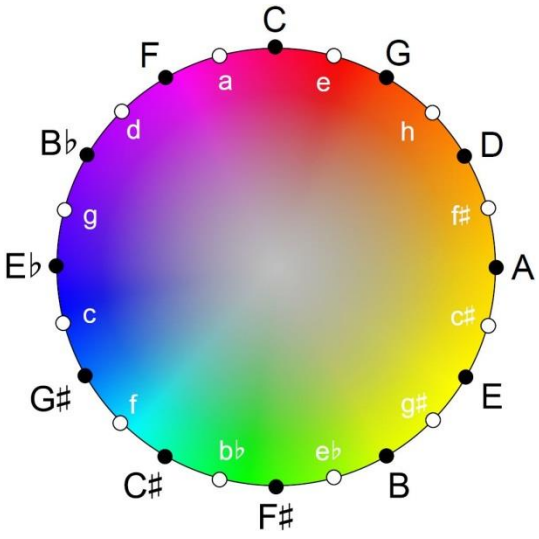


Figure 42: Circle of Thirds [46]

Even though the visualization presented by [46] attempts to display music theory using colour theory. It is a unique visualization because it aims to be a hybrid system that is aesthetically pleasing and easy to understand for analysis. Unfortunately the authors did not conduct user studies to validate the visualization system.

2.2.2.3.1.1 Psychological Models of Music

In order to convey the emotional and structural information of music to the D or HOH through visualizations, auditory cues must be translated to the visual medium. However, there are two problems to overcome.

The first problem rests in the method of translating auditory constructs to visual constructs. The issue with the translation is that the auditory and visual systems in the human brain are separate mechanisms and circuits, which have very little overlap [27]. This means that many of the emotions evoked by the auditory system are separate from the visual system, and vice versa. It is very difficult to evoke the same emotions that music triggers when using the visual system to represent music.

The second problem is that music is often categorized in eight discrete high-level emotions: (1) anger, (2) fear, (3) sadness, (4) disgust, (5) surprise, (6) anticipation, (7) trust and (8) joy [47], which is an over simplification. The emotions evoked by music often do not fit into one category but may be a part of two conflicting categories such as sadness and joy. Conflicting emotions such as sadness and joy are possible. For example, in German, Schadefreude is an expression directly translated as *Shameful-Joy*. Also, an individual's prevailing mood may dictate the emotion of the music [21], hence musical emotions are influenced by the interpretation of the individual. Creating a happy visualization based on a happy song is difficult because perceived musical emotions are based on individual interpretations. Also musical pieces that are generally described as sad have very similar pitch, tempo and rhythm constructs in relation to songs that are considered fearful [2]. Therefore classifying music in discrete categories automatically is very difficult for certain songs, where multiple conflicting emotions are triggered.

Music visualizations described in the previous section utilize colours for emotional transfer. This poses several challenges since there is no set of common meanings for a given colour. Most colours have positive and negative meanings that not only differ among countries but also among individuals within the same culture [3]. [3] states that, in the Western World, black can represent sexuality while representing a state of mourning. Another representation of sexuality is red, whereas red can also represent war, blood, communism, danger or stop. Also in Western cultures, white represents purity and maidenhood in weddings, yet in India, it represents death. Considering colours are perceived differently among individual cultures and among people, they cannot be relied upon to represent any emotional information in a music visualization.

In order to create an informative and entertaining visualization, MusicViz does not attempt to interpret emotions conveyed in the musical pieces due to the challenges stated by [21], which states that one's prevailing mood determines to an extent the types of emotions a song elicits. Instead, it relies on psychological translations of musical structures into pre-attentive visuals. This allows each individual to interpret the visual translation based on their own previous experiences, which is the same process hearing people follow for auditory music. Even though music evokes numerous emotions that differ from song to song, all music is a combination of pitch, tempo and volume [2, 14]. MusicVis translates these basic musical structures, which are common to all music, into the visual medium. MusicViz does not attempt to interpret the emotions in auditory music. It only translates musical structures into pre-attentive visual structures, which deaf and hard of hearing individual can interpret.

Pitch, tempo and volume are shown in [2] to be the carriers of the three affective dimensions: valence (pleasant-unpleasant), energy arousal (awake-tired) and tension arousal (tense-relaxed); they are the transmitters of emotional information. In addition, each musical construct interacts and is able to affect the other, which creates more complex emotions. By manipulating valence, energy arousal and tension arousal, emotions such as sadness and fear can be differentiated, which are comparable in valence and energy arousal but differ tension arousal [2]. In relation to cross interaction between the three affective

dimensions, valence ratings are affected by volume, tempo and pitch (low volume has higher valence and fast tempo with lower pitch height has higher valence); energy arousal is affected by volume and tempo (high volume and fast tempo has more engery); and volume, tempo and pitch affect tension (high volume, fast tempo and high pitch has higher tension) [2]. Considering the three affective dimensions are important facets into conveying emotion in music, MusicViz uses visual associations to create an emotion based visualization system.

Each of the three affective dimensions has translations that are pre-attentively understood in the visual domain: (1) volume is associated in two different ways in the visual domain. First, volume is associated with visual brightness; the louder a sound is the brighter it is [14]. The second affiliation of volume is the psychological reaction to looming objects. Both auditory and visual looming cues indicate an evolutionary response to fast approaching danger in order to avoid them [48], hence larger objects indicate a louder sound. (2) Pitch height is associated with colour brightness, object size and vertical height. A higher pitch is psychologically linked with a brighter colour, whereas a lower pitch is linked with a darker colour. A higher pitch is also affiliated with smaller objects, while a lower pitch is linked with larger objects. (3) Lastly, different tempos are shown to be strongly associated with evoking emotions in the visual and auditory domain. Faster tempos are intuitively shown to create higher emotional arousal whereas slower tempos trigger calmer emotions [49].

Even though the three affective dimensions firmly convey pre-attentive emotional information when viewed independently in studies, problems arise when they are combined into a single visualization. The issue is that both pitch and volume are linked to volume and size; therefore, pitch and volume cannot alter the same variables, without causing confusion. This is the same issue that information visualization experiences. For example, different colours and different shapes are pre-attentively understood independantly; however, a visualization combining a different colour with a different shape is not.

2.2.2.3.1.2 MusicViz's Psychological Model

MusicViz overcomes the problem of combing pitch, volume and tempo into one visualization by utilizing three-dimensional spaces. This provides the visualization with the X, Y and Z-axes as extra variables for mapping pitch height, tempo and volume. The information that is visualized is based on the score of the musical performance, otherwise referred to as sheet music for musicians. The MIDI file standard provides this information in real time through MIDI messages.

A MIDI message, at a general level, is composed of the role it plays in the MIDI system and its target channel. The MIDI standard allows for 16 channels, where each channel represents a family of similar instruments. For example, a channel can represent string instruments and contain two guitars. The channel setup is unique to each song; however, the 10^{th} channel is always reserved for percussion instruments. For the purpose of the visualization, the following three messages are used: (1) note on/off messages (channels 1 to 9 and 11 to 16), (2) percussion messages (channel 10) and (3) pitch-bend messages.

Note on/off messages are composed of three bytes. The first byte signifies the target channel and whether it is a note on or off signal. The second byte indicates the note (pitch) played and the third byte communicates the volume at which the pitch is played. Considering the volume and pitch are represented by bytes, the range of values is from 0 to 127. Percussion messages are constructed in the same fashion as note on/off messages. The difference is that any note on/off message which targets channel 10 is only for percussion. Pitch-bend messages target all notes played in a channel and "bend" them by either increasing or decreasing them by a certain amount, such as the function of a whammy bar on a guitar.

MusicViz is designed to parse the incoming MIDI messages just described and translate the data into the proper visual representation. The program is designed in Java 6 and utilizes 3D graphics through JOGL 2.0 (Java OpenGL). JOGL 2.0 provides C style OpenGL calls by acting as a wrapper class for OpenGL functions; hence the functions are accessed in the same procedural and state-machine oriented structure as

in C and C++. Please see the System Design and Methodology chapter for an in depth technical discussion.

In order to represent music in a visual manner, MusicViz represents non-percussive instruments as coloured pipes, which is inspired from the Windows 95 pipes screen saver. The position and size of the pipes are altered as the song progresses. These changes convey pitch changes, volume changes and tempo. Considering each of the 16 MIDI channels represent an instrument family, MusicViz assigns two pipes to every channel (besides the 10^{th} channel) to convey the pitch, volume and tempo changes in the channel. Different structures are needed to represent percussion. Like all instrument families, they have their own unique timbre; however, a physical generic drum is visually identifiable by deaf audiences due to its unique look and prominence in deaf music. Therefore, MusicViz uses five large toroid objects to represent percussive instruments such as the bass drum, toms, snare drums and high hats.

As described in the previous section, both pitch and volume are pre-attentively understood in the visual system when mapped to colour brightness and size; however, the problem is that both musical constructs cannot occupy colour brightness and size concurrently. The solution is to use the y-axis in 3D space to represent pitch height and allow brightness and size to represent volume. This means that pipes with higher pitches are placed higher on the y-axis of the 3D world and brighter colours and larger pipes represent louder notes.

The x-axis is utilized by placing the pipes at pre-determined values along the x-axis. The pipes that belong to the same instrument family are placed at the exact same starting position on the x-axis; however, they are stacked on top of each other. This helps group similar instruments into identifiable gatherings. To further help instrument groupings, MusicViz assigns colours to instruments in the same family. It is important to note that colour is never used in the visualization to evoke emotions. This is due to difference in cultural and personal interpretations [3]. Lastly, MusicViz utilizes the z-axis to show the

flow of time and tempo for the musical piece. Table 1 summarizes the auditory translations into the MusicViz program.

Table 1: MusicViz's audio to visual mapping

Auditory Construct	3D Translation to MusicViz
Pitch	Pipe's y-axis position
Volume	Pipe size (diameter) and colour brightness (alpha)
Tempo	Pipes x-axis position

2.3 Vibrotactile

As discussed in the MusicViz section, MusicViz is one part of a two part music experience system. The second part consists of the Emoti-Chair to provide the vibro-tactile information. However, before discussing the Emoti-Chair as a hardware device, the Emoti-Chair's background theory must be described. This section describes the device's background theory, such as the physiological and auditory experience of vibration as well as the use of vibration as a sensory substitution technique. In addition, various vibro-tactile sensory substitution techniques are discussed that present speech and visual information as vibrations using various devices. Finally two haptic feedback chairs are presented that aim to communicate musical information. The chairs utilize theory and physiology of vibration to communicate the information. One of the chairs is the Emoti-Chair created at the IMDC and the other chair is created at the National University of Singapore.

2.3.1 Physiology and Frequency Theory

Western music utilizes a 12-tone octave tuning system [50]. This is referred to as a perfect octave. An interval between two musical pitches is half or double the note's frequency; hence, the ratio between octaves is 2:1. For example one octave above A4 (440Hz) [50] is 880Hz and one octave below is 220Hz [50]. Octaves with a distance greater than 1 are calculated as 2^n and the reciprocal of the series, where n denotes the octave [50]. Perfect octaves are further split into 12 semi-tones, where each semi-tone contains 100 cents; hence, each octave contains 1200 cents [50]. The amount of cents between two notes is calculated using the equation below (see Equation 1). Therefore the frequency 440Hz and 880Hz are separated by 1200 cents, which is one perfect octave. See Equation 2 in order to calculate a note of n octaves away. For example, the note residing 6 semitones away from 440Hz is 622.25Hz

Equation 1: Calculate the number of cents *n* between note frequency *b* and *a*

$$n = 1200 * \log_2(\frac{b}{a})$$

Equation 2: Calculate the frequency of a note *b* that is *n* cents away from note *a*

$$b = a * 2^{\frac{n}{1200}}$$

[50] states that humans can differentiate auditory pitches separated by 5 to 6 cents and musically untrained adults can reliably recognize differences of 25 cents [51]; however, pitch sensitivity is dependent on timbre and harmonics of the note. Notes produced by instruments are not simply a single frequency; they are "a mixture of sinusoids of different frequencies with different weights that are perceived to be equivalent to single frequency" and are referred to as a note's fingerprint or timbre [50]. The note's timbre reduces a human's ability to differentiate pitches by +/- 12 cents [52].

There exists previous research into translating audio to vibro-tactile information. In 1924 Gault experimented with the capacity of the body to present speech information as vibro-tactile information to the skin [53]. The study resulted in artificially deafened participants being able to recognize speech through vibrations [53]. In 1932 Alcorn [54] presented a method of communication for use by the deaf and blind. An individual places their thumbs on the speaker's mouth and fingers on the speaker's jawline [54]. These methods of presenting tactile information focus only on communicating speech information. Speech signals lack the complexity of music signals because many frequency signals combine and overlap to create complex sounds such as timbre. Timbre as stated above is a mixture of waveforms of different frequencies and weights that are perceived as a single frequency.

[55] states that music can be expressed through the vibro-tactile domain; however, using a simple actuator cannot accurately depict the wide range of audio signals in music. Expressing music through the tactile domain, as through the auditory domain, relies on the ability to discriminate frequency changes. [55] states that most vibro-tactile research examined frequency discrimination using single contactors of sizes less than 15mm such as the research in [56-58]. The actuators in these studies are placed on the participants' arms [55]. [58] showed that participants' ability to distinguish separate frequencies increased as the difference between frequencies increased between 25Hz and 200Hz using a 6.5mm actuator. [59] shows that impairing glabrous skin using an anesthetic prevented the detection of 20hz and 50Hz frequencies; however, frequencies from 100 to 300 are easily differentiated. [57] shows that there are 7 differentiable steps between 100 and 700Hz using a 14mm actuator. [56], utilized a 2.5mm actuator, discovered seven differentiable steps between 10hz and 90hz for actuators on the forearm and 10 differentiable steps on the finger.

Unlike the previously described papers, which present vibro-tactile stimulation to the users arm and/or fingers, [55] presents research into the FDL (Frequency Difference Limens) of actuators bigger than 100mm along the user's back. In addition [55] presents a frequency discrimination paradigm to discover the frequency resolution of notes common to music. The back is utilized for four reasons: (1) frees the

user's hands for other tasks [55]; (2) humans are familiar with vibrations from car engines and other stimuli through the back [55]; (3) the back is a large surface area allowing for spatial separation of vibrations [55]; (4) the back can be used in a seating position allowing for comfortable use of haptic devices [55] that allows for long term use by the hard of hearing and deaf to access music.

Branje (2010) states that participants are able to distinguish differences between frequencies as low as 200 cents and consistently distinguish frequency differences below 400 cents (1/3 of an octave) [55]. These results show that the back is not as sensitive as the finger and forearms, which are able to recognize differences as low as 10Hz. This is because the back is inherently less sensitive than the arms; however, Branje (2010) states that sensitivity improvement is proportional to the actuator's size. Even though the back is shown to be able to distinguish frequency differences of 200 cents, this is still less than the auditory system, which distinguishes differences as low as 6 cents [50].

2.3.2 Vibration as Sensory Substitution

Sensory substitution is a large multi-discipline field spanning medicine, computer science, engineering, psychology, etc. Sensory substitution systems provide users with information through a human sensory channel that differs from the traditional input [60]. [61] shows that braille reading by blind individuals stimulates the occipital lobe in the brain as well as shows that the poor representation of locomotor space in early blind individuals improved with the use of haptic sensory substitution devices [61]. The authors trained totally blind individuals to use an ultrasonic echolocation device over multiple 1-hour training sessions [61]. The device is a substitution system consisting of a pair of spectacles with three ultrasonic transducers, which communicate distance and direction of objects. The results show that, through brain plasticity, the occipital lobe is recruited in new physiological activities that improve locomotor representation [61]. In addition the same phenomena are shown to occur in blind cats, where the visual cortex is recruited for auditory and tactile localization [62]. These cats are shown to have few overt impairments that negatively affect their natural behavior [62]. [22] presents a similar sensory substitution

device that allows blind individuals to see using a camera that translates the information to electrical impulses. The electrical impulses are sent to a tactile electrode array on the tongue [22].

Other techniques for haptic feedback for use by the blind are presented in [63]. The Optacon is a reading aid that translates physical documents into 144 black and white haptic dots displaying the dots on braille screen. Blind users experiences reading speeds up to 100 WPM [63]. The Optohapt (not to be confused with the Optacon) translates words to a series of signals that are displayed on different parts of the body [63]. TVSS is designed to transmit visual pictures to an array of 400 vibrators on the user's back. Blind users can recognize pictures with this device [63]. Another device is the Kinotact, which is similar to the Optohapt. Instead of using 400 vibrators, it utilizes a 10x10 vibrating array, where each vibrator corresponds to a photocell [63]. Blind users are shown to recognize images [63]. The Felix system is the precursor to the modern hearing aid [63]. The Felix system utilizes a harness, which translates audio from a microphone on the user's belt to a series of vibrations on the user's arm [63]. Devices such as the OMAR system translate the tactile and kinesthetic senses [63]. The OMAR system is able to encode vibration and movement such that speech information is encoded and transmitted to a user's finger [63]. Another device is a reverse Morse code machine that moves a blind user's finger up and down to communicate words [63]. User studies show that blind individuals can comprehend up to 20 WPM.

In addition, sensory substitution techniques are implemented in robotic surgical systems. [64] presents research into haptic feedback mechanisms for suture-manipulation. [64] states that even though robotic surgical systems are successful, they are not successful in the cardiovascular field because of space restrictions from the ribcage. In addition, traditional systems lack haptic feedback preventing surgical tasks from being performed on highly delicate tissue [64]. Surgeons use pressure and haptic feedback to create surgical knots that are tight enough to tie delicate tissue together without harming tissue [64]. The results from [64] suggest that haptic feedback during suture tying with robotic devices are superior to that achieved with hands alone.

[65] presents a similar problem, where the lack of haptic feedback creates errors in human interaction with virtual objects. The real world provides individuals with multiple modes of sensory input. When interacting with objects, friction and other kinesthetic and tactile information provide feedback for delicate or rugged manipulation [65]. However, because virtual environments lack haptic feedback, users make erroneous judgments when interacting with virtual objects [65]. The solution proposed in [65] solves the haptic feedback problem through the use of a haptic feedback glove that imitates pressure from manipulating an object. The results show that users' performance increased up to 61.9% when using haptic feedback and visuals [65].

The research previously described in this section focuses on sensory substitution model and devices that either communicates location information or words to blind individuals; they do not offer any methods to translate music to the vibro-tactile domain for the deaf or hard of hearing. In addition, Karam (2009) states that there exists "little work in the area of music enjoyment, towards making the emotional experiences of music more accessible to deaf and hard of hearing people" [66]. The research from Branje's (2010) directly contributes to the development of the Model Human Cochlea (MHC) [55]. The MHC is a sensory substitution technique based on the human cochlea that presents music as multiple discrete vibration channels on a haptic display [55, 66, 67]. The goal of the MHC is to translate the emotional content of sound from music and film onto a haptic display to enhance the user's entertainment experience [67]. Therefore the MHC is unique because it does not aim to aid in information communication; instead it aims to provide an entertaining vibro-tactile experience of music and movies.

The deaf and hard of hearing already seek musical experiences through close contact with audio speakers; this enables individuals to feel the music. However, masking issues exist with this form of accessing music. Higher frequencies are masked by the lower frequencies, resulting in the higher frequencies, which are of lower power to be drowned out by the lower frequencies, which are of higher power. It is through these low frequencies that the hard of hearing and deaf experience music. This is apparent in deaf music and deaf culture. Deaf individuals go to clubs and live concerts to experience music, where the bass

component is strong. In addition, deaf culture is not devoid of music; the deaf create music through rhythmic gestures put to bass [10].

The MHC model mitigates the masking problem by isolating frequencies from musical recordings and haptically displaying the frequencies as multiple discrete bands spatially separated along the body [66]. The MHC is based on and uses the human cochlea as a design metaphor [55, 66-68]. The human cochlea separates "sounds according to their frequency" [69]; hence, the human cochlea contains a tonotopic ordering of hair cells that vibrate when a specific frequency is observed [66]. Karam (2009) states that audio and tactile signals are similar; both signals are measured in terms of frequency and amplitude. In addition, audio and tactile signals are detected through vibrations. Using the human cochlea as a model, the MHC spatially separates frequencies and presents vibrations along the user's back using an array of 8x2 actuators [66].

Karam (2009) presents user tests into the effectiveness of utilizing the MHC as a means to transmit auditory as haptic vibrations. Twenty individuals from an old age home used the chair; however, only 7 participated in the study. The results show that all but one participant used their fingers to dance with the vibtracks [66]. Three out of the four deaf women expressed accurate facial expressions that reflected the music [66]. Others used gestures to pretend playing violins and using a conductor's baton. In conclusion the MHC model proves to be reliable at transmitting emotional information of music to deaf audiences through haptic display, which spatially separates frequencies and presents them along the user's back.

2.3.3 Haptic Sensory Substitution Devices

The MHC, as proposed in [66, 67], is utilized in the Emoti-Chair [68, 70-72]. The original Emoti-Chair consists of a gaming chair with sixteen voice coils, a low frequency actuator and four air-jets [70]. The authors utilize four subwoofers with a frequency range of 55Hz to 2,500Hz [70]; a low frequency transducers with a frequency range of 20hz to 80hz [70]; five voice coils with a frequency of 500hz to 15,000Hz [70]. All together the preliminary Emoti-Chair provides 9 independent audio channels [70],

where the voice coils and subwoofers are placed on the arm rests, back and seat [70] and four air-jets blew air onto the users.

This model is no longer used due to the complexity of the set up. It required multiple amplifiers and controller boards to drive the air-jets, actuators and subwoofers. The current design of the Emoti-Chair (see Figure 43) [71, 72] does not contain subwoofers, controller boards and air-jets. Music input to the chair is split among the chair's eight channels (an array of 8x2 actuators). Each channel contains two actuators and is placed on the user's back. The Emoti-Chair accepts music input in two forms. The first input form is through wave-form based signals (MP3, WAV, and MP4). When a waveform is input to the system, it is frequency segmented into eight unique frequencies between 27.5Hz and 1,000Hz. Each channel is assigned a specific frequency range to play. The low and high frequencies are mapped to the bottom 4 and top 4 channels accordingly. This frequency distribution is based on the MHC described above. The second method of input is to directly map instruments to channels. This is useful for live musical performances, where output from instruments can be utilized. This thesis utilizes the Emoti-Chair as the haptic feedback component of the VITA system and utilizes the second method to map audio to the tactile domain.

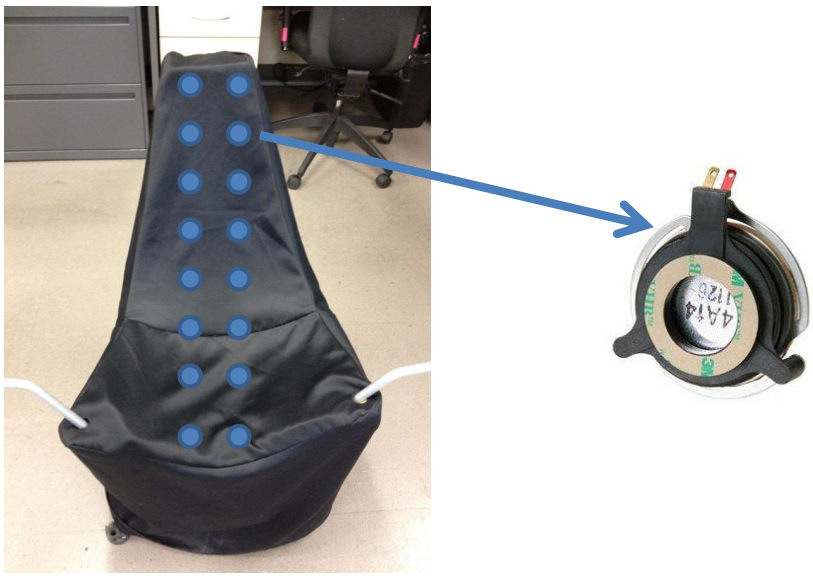


Figure 43: Current Emoti-Chair and actuator positions

The Emoti-Chair is not a stand-alone device. In order to power the eight channels, two pyramid audio amplifiers are used (see Figure 44). Each amplifier powers four channels. The amplifiers drive the actuators by specifying the frequency and amplitude of each channel. The Firepod receives the frequency and amplitude information. The Digital Audio Workstation (DAW) is directly connected to the computer to facilitate communication to the amplifiers. This thesis uses the PreSonus Firepod (see Figure 45) as the DAW.

Figure 44: 300 watt Pyramid Amplifier

Figure 45: Presonus Firepod

Branje [72] conducted user studies to gauge the emotional capacity of the Emoti-Chair. A total of 59 individuals participated and were presented with video clips of a horror movie; each individual's galvanic skin response (GSR) was measured during the task. [72] states that an increase in skin conductance levels coincide with levels of emotions [73, 74]. The authors presented the participants with four video clips, where two clips were of a startling scene in a film and the two other were suspense clips. The user study in [72] suggests that the Emoti-Chair can convey some emotional aspects of film. The average change in skin conductance levels between the rest period and the presented horror video are highest when the audio, video and haptics were presented together [72]. Even though user studies were conducted, a limitation is that no hard-of-hearing or deaf individuals were included [72].

[15] presents a sensory substitution system composed of a music visualization-described in the previous section- and a haptic feedback chair [15]. The authors realize that vibrations are used by D musicians and individuals to experience music. Hence, the enjoyability of music can increase if sound is amplified

through the body without adding artificial effects or changes to the audio signal [15]. Four speakers are attached to an Ikea 'Poäng' chair. The 'Poäng' chair is a bent beech wood chair that has layers of wood glued together [15], which allows users to easily rock the chair. In addition this allows vibrations to travel unhindered through the material [15]. Two speakers are mounted to each of the arm rests, one speaker is mounted at the lumbar spine and one is mounted to a floor-board to conduct vibrations through the feet [15]. Dome-like hand rests are used to amplify high-frequency vibrations through the forearm and fingers of the users. The haptic feedback chair is shown in Figure 46.

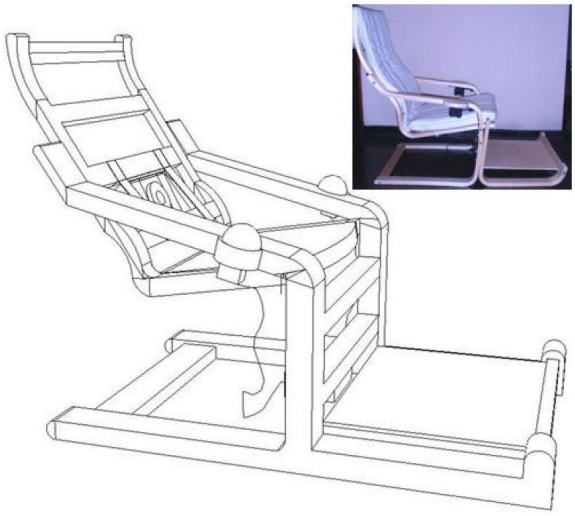


Figure 46: Haptic Chair [15]

User tests were performed to test the effectiveness of the visualization and haptic feedback system and the results were positive [15]. Participants were encouraged to be relaxed as they would be at home and were allowed to use music on their MP3 players [15]. One participant stated that "there is no different between me and a normal hearing person. I feel proud" [15]. Profoundly deaf individuals stated they could "hear" the music and "hear from their legs" [15].

2.3.4 Thesis Outline

The thesis is structured as follows:

Chapter 1: provides an introduction to the thesis and gives an overview of the system, what the system is supposed to accomplish, background information and the purpose of the user study.

Chapter 2: is the literature review. This chapter describes the history of visualization and the history of music visualizations from their purpose and origin in theatre and plays to their modern renditions using animation and more recently computer animation. The music visualization program designed for this thesis is described as well as the various psychological models pertaining to the visual perception of auditory cues that are used as the basis for MusicViz's translation principles. The chapter also provides a background into the physiology of tactile feedback and the use of haptic feedback devices for the use of communicating music tactically.

Chapter 3: describes the architecture and design of MusicViz and how it is coupled with the Emoti-Chair to create VITA. This chapter provides detailed descriptions of the implementation of MusicViz's translation principals for rendering in 3D graphics. This chapter also describes the software architecture of the Emoti-Chair.

Chapter 4: describes the usability study conducted to test the viability of VITA as a cross-modal sensory substitution system for the use by D and HOH individuals to access music for entertainment purposes. This includes the study procedure as well as the study's results, findings and discussion.

Chapter 5: describes the conclusion, limitations of the thesis as well as the future work that can be conducted on the VITA software.

3 System Design and Evaluation Methodology

3.1 Overview

The system designed for this thesis is named Visually Immersive and Tactile Animation (VITA). VITA is comprised of two separate and different systems with which the user interacts. One system is the MusicViz software system developed for this thesis and the other is the Emoti-Chair [71, 72] hardware

system. MusicViz is a music visualization system that translates auditory constructs such as pitch, amplitude, and tempo from a standard MIDI file into 3D visual constructs on the basis of a psychological model that provides a perceptual mapping of these auditory constructs into the visual domain. This occurs in real-time. The Emoti-Chair is an existing system that provides vibrotactile feedback chair that translates auditory music to tactile vibration along the user's back and is described in Section 2.3.3 of the literature review. VITA allows for MusicViz to playback a MIDI sequence on a computer screen as well as on the Emoti-Chair. Figure 47 shows a general overview of the VITA architecture.

Section 3.2 describes the libraries and technologies used for MusicViz's development. This includes the relevant MIDI data, Java's MIDI playback structure, OpenGL and MusicViz's visualization algorithms. Section *3.3* describes the method in which MusicViz and the Emoti-Chair are connected to create VITA. The other three sub-sections comprise of the methodology (Section 3.4), results (Section 4.1) and the discussion (Section 4.2). The methodology section describes the VITA's evaluation method, such as the experiment procedure, research questions, participant demographics and study results.

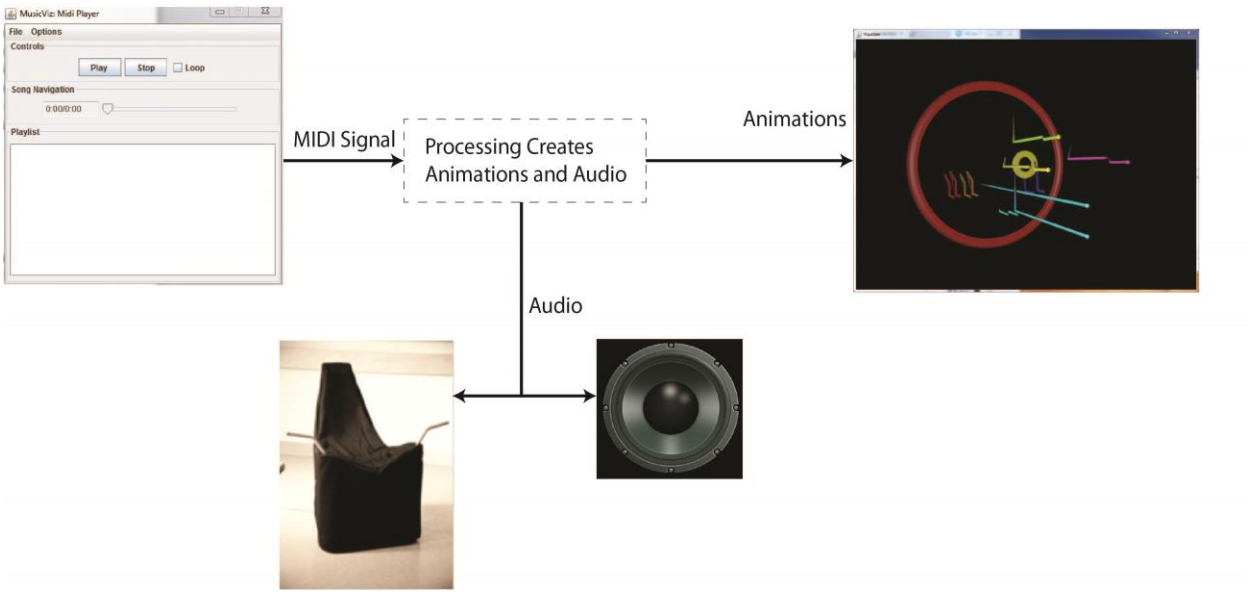


Figure 47: General overview of VITA

3.2 MusicViz Software

MusicViz is programmed in Java 6 and uses JOGL 2.0 (Java OpenGL) release candidate eight [75] for 3D graphics.

As described in the *Literature Review*, automated music visualization systems use basic software algorithms to extract beat information to render in a visually appealing manner. However, relevant audio information must be translated from the auditory domain into the visual domain to create an emotional, entertaining and informative experience. The results from [10] indicate that, by creating a translation algorithm based on the 3D model of emotion, MusicViz can create an entertaining and informative music visualization. The 3D model of music represents music using valence (positive-negative), arousal (awake-tired) and tension (tense-relaxed) [10]. [2] shows that valence ratings are affected by volume, pitch and tempo; arousal is affected by volume and tempo; and tension is affected by volume, tempo and pitch.

Pitch, volume and tempo have equivalent visual representations that are shown to be pre-attentively understood. Volume is linked with visual brightness [14] and the psychological effect to looming objects, where looming queues cause an avoidance response [48]. High and low pitches are psychologically linked to bright and dark colours respectively [49]. Pitch is also associated with object size, where a higher pitch is associated with a smaller object and a lower pitch is associated with a larger object [49]. Lastly, fast tempos are shown to intuitively create higher emotional arousal and slower tempos elicit calmer emotions [49].

Even though pitch, volume and tempo are shown to be pre-attentively understood when viewed individually, a problem arises when they are combined. The problem is that both pitch and volume are linked to brightness and size [10]. [10] suggests that there is confusion when both constructs alter the same variables. This is addressed by utilizing 3D space for the three dimensions of music. The translation algorithm is summarized in Table 2. Alpha refers to the transparency in the RGBA colour model used in computer graphics. The pipes are shown (Figure 48).

Table 2: MusicViz's audio to visual mapping

Auditory Construct	3D Translation to MusicViz
Pitch	Pipe's y-axis position
Volume	Pipe size (diameter x-axis) and colour brightness (alpha)
Tempo	Pipe's z-axis position

3.2.1 MIDI Standard and Java MIDI API

Java's API provides built-in objects to handle MIDI operations. Such operations include MIDI playback and recording through software sequencers. The API also provides object representations of MIDI data. A GM (General MIDI) message is comprised of its role in the MIDI system and its target channel. The GM standard allows for sixteen channels, where each channel represents a set of similar instruments [76]. The channel setup is unique to each song; however, the tenth channel is always reserved for percussion instruments [76]. MusicViz uses the following three messages as the base for its auditory-to-visual translation algorithm: (1) note-on/off messages (channel one to nine and eleven to sixteen); (2) percussion messages (channel ten); and (3) pitch-bend messages (see Table 3 for the detailed MIDI messages used in MusicViz).

Table 3: Relevant MIDI messages to MusicViz [77]

Status Byte		Data Bytes	
1st Byte Binary|Hex|Decimal	Function	2nd Byte	3rd byte
1000 0000=0x80=128	Channel 1 Note off	Note Number (0-127)	Note Volume (0-127)

…	…	…	…
1000 1111=0x8F=143	Channel 16 Note off	Note Number (0-127)	Note Volume (0-127)
1001 0000=0x90=144	Channel 1 Note on	Note Number (0-127)	Note Volume (0-127)
…	…	…	…
1001 1111=0x9F=159	Channel 16 Note on	Note Number (0-127)	Note Volume (0-127)
…	…	…	…
1110 0000=0xE0=224	Channel 1 Pitch Bend	Pitch Bend LSB (0-127)	Pitch Bend MSB (0-127)
…	…	…	…
1110 1111=0xEF=239	Channel 16 Pitch Bend	Pitch Bend LSB (0-127)	Pitch Bend MSB (0-127)

3.2.1.1 Note-On/Off Messages for Percussion and Instruments

Note-on/off messages are composed of three unsigned bytes [77]. The first byte (status byte) signifies the target channel and whether it is a note-on or note-off signal. The second byte indicates the note (pitch) played. The third byte communicates the volume at which the pitch is played. Percussion messages are constructed in the same fashion as note-on/off messages. Any note-on/off message targeting channel ten is only for percussion [76].

Specifying the channel, the pitch and the volume creates a note-on message. An example of a note-on message for channel 11 with a pitch of 127 and a volume of 100 is shown in Table 4. When a note-on is triggered, the MIDI synthesizer continuously plays the note at the volume and pitch specified until a corresponding note-off message is received. There exists one technique to create a note-on signal. However, there are three approaches to create note-off messages (see Table 4), where each method is semantically equivalent. The first method creates a note-off message with a zero volume. The second method creates a note-off message with a non-zero volume. The third method creates a note-off message

using a note-on message with a zero volume. MusicViz converts each note-off signal, which is specified using method 2 and 3, to a note-off signal using method 1. This allows for consistent processing of MIDI note-off messages.

Table 4: A note-on message targeting channel 11 and the three methods of specifying a note-off equivalent

Function		Status Byte	Pitch Byte	Volume Byte
Note-on		154	127	100
Method 1	Note-off with zero volume	138	127	0
Method 2	Note-off with non-zero volume	138	127	100
Method 3	Note-off created with zero volume note-on	154	127	0

3.2.1.2 Pitch-Bend Messages

Pitch-bend messages are similar to note-on/off messages but target all notes played in a channel instead of triggering a specific instrument playing a specified pitch and volume. Pitch-bend messages alter all notes in the channel by either increasing or decreasing the pitches. This function is analogous to a guitar's whammy bar. See Appendix 6.12.1 for more information on pitch-bend messages.

3.2.2 Visual Structures in MusicViz

In order to represent music in a visual manner, MusicViz uses three dimensions to perform the auditory-to-visual mapping. MusicViz represents non-percussive instruments as coloured pipes (see Figure 48), which is inspired from the Windows 95 pipes screensaver [36]. The position and size of the pipes are altered as the song progresses. These changes convey pitch changes, volume changes and tempo. The MIDI standard contains sixteen channels, where each channel represents an instrument family. MusicViz assigns two pipes to every channel (except for the percussion channel) to convey the pitch, volume and tempo changes.

Different structures are needed to represent percussion. A physical drum is visually identifiable by D audiences due to its unique look and prominence in D music. MusicViz uses five toroid objects (see

Figure 48) to represent percussive instruments due to the similarity to physical drums. The tenth MIDI channel represents percussion.

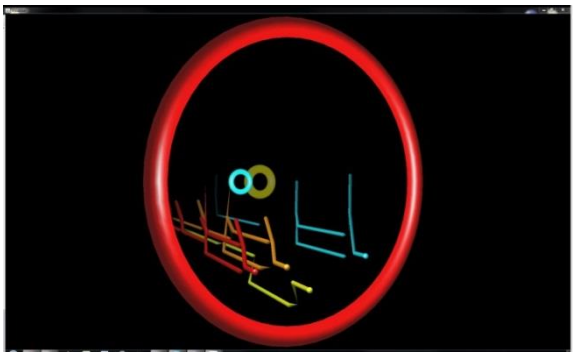


Figure 48: MusicViz visualization (pipes for instruments and toroid objects for percussion)

As seen in Figure 48, the x-axis is used to place placing the pipes at equally spaces values. Pipes belonging to the same instrument family are placed at the same starting position on the x-axis; however, they are stacked on each other. This helps group similar instruments into identifiable gatherings. To further group instruments, MusicViz assigns a unique colour to each MIDI channel. It is important to note that MusicViz does not use colour to evoke emotional responses rather colour is used to visually show similar instrument families.

Pitch is represented by a pipe's position on the y-axis. Lower pitches appear lower on the screen and higher pitches appear higher on the screen. MusicViz uses the z-axis to show the flow-of-time and tempo for the musical piece. The translations are summarized in Table 2. Even though Figure 48 shows three toroid objects, the MusicViz system can represent drums with up to five toroid objects. The groupings are shown in Appendix 6.14.2.2.

3.2.3 MIDI Sound Playback Structure

MusicViz exploits Java's MIDI API for MIDI playback by intercepting MIDI messages during playback. MusicViz uses this feature to create real-time music visualizations. Java's MIDI library contains an *Interface* called *Receiver* and a class called *Sequencer.* The *Sequencer* class, when initialized, has the

ability to playback a MIDI song and generates audio using a *Synthesizer* object. The *Receiver* interface allows MusicViz's objects to connect to Java's *Sequencer* and intercepts each MIDI message played (see Figure 49). The MIDI messages are translated in real-time to their visual equivalents and then are animated. For more information, see Appendix 6.12.2.

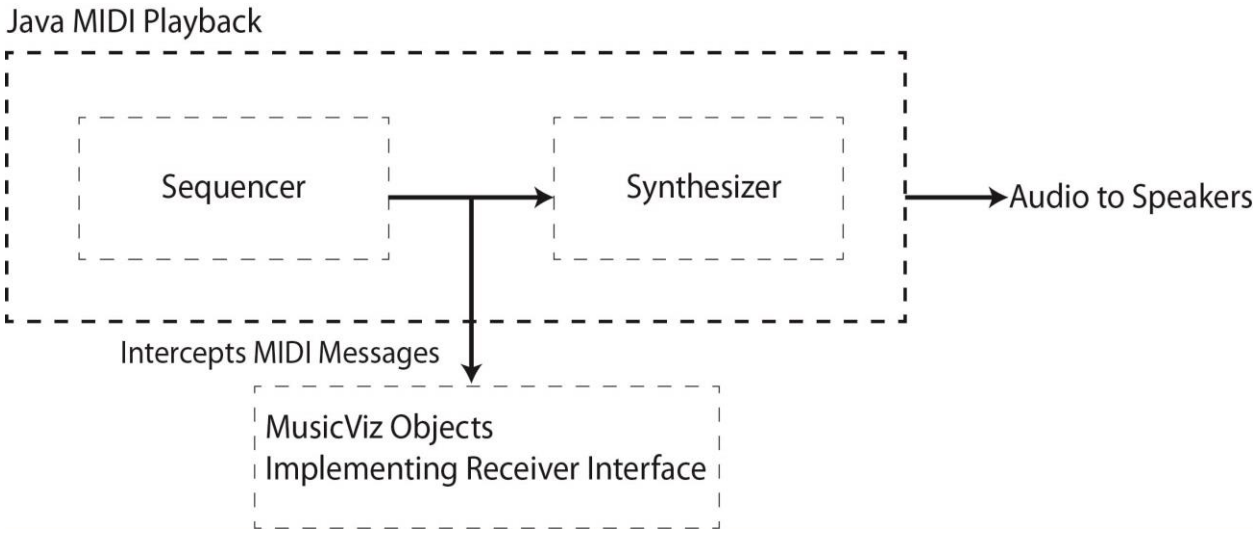


Figure 49: Overview of how MusicViz's objects implement and intercept MIDI messages in real-time

3.2.4 MusicViz Program Initialization and OpenGL Structure

Before discussing the visualization algorithm and the associated system architecture, it is important to discuss the graphics and GUI initialization.

MusicViz utilizes JOGL 2.0 [75] for OpenGL graphics. JOGL is a wrapper library containing wrapper classes for OpenGL's use in Java. JOGL allows for C-style OpenGL calls through the Java Native Interface library. This allows the programmer to make native C-calls from Java without performance degradation. In addition, JOGL provides all GL and GLU calls that OpenGL in C provides. Java also provides two windowing systems (Swing and AWT) for Windows, Mac and Linux.

During MusicViz's initialization, two windows are created for the user. The first is the *MIDI player* designed in Java's SWING API. This window contains controls for MIDI playback, such as stopping, pausing, seeking and loading MIDI songs (see Figure 50). The window also contains the functionality to

control the visualization. These settings are found within the *Options* menu. The *Options* menu allows the user to choose whether the pipes or percussion are displayed on screen and played through audio. The second window displays the *OpenGL visualizations* (see Figure 48). This window is designed in Java's AWT API. The *MIDI player* window is the input to the system and the *OpenGL window* is the output.

Figure 50: MIDI Player GUI Controls

The initialization of the two windows occurs in *GUI.java* within the *startPlayerGUI* method and the *startVisualizer* method (see Appendix 6.7). The important process, shown in Appendix 6.13, during initialization is creating the visualization components. The pipes of MusicViz are preloaded into graphics memory during initialization using vertex buffer objects (VBOs) which are then altered during animation.

VBOs load various data types into video memory for non-immediate rendering. Such data include vertex data, colour data, vector data and normal data. VBOs are low-level constructs that directly access memory in a linear fashion without high-level data structure abstraction.

The pipes in MusicViz are represented in OpenGL as a set of geometrical faces spaced apart from each other (see Appendix 6.13) and are stored using VBOs. For the detailed data structure implementation and storage of the pipes (see Appendix 6.13).

3.2.5 MusicViz Playback and Visualization Architecture

The architecture of loading a MIDI sequence, receiving, processing and visualizing MIDI messages is divided into four stages in MusicViz. One or more threads handle each stage. Data communication between threads is handled by thread-safe data structures. Figure 51 shows the four processing stages and their associated threads within MusicViz's translation algorithm.

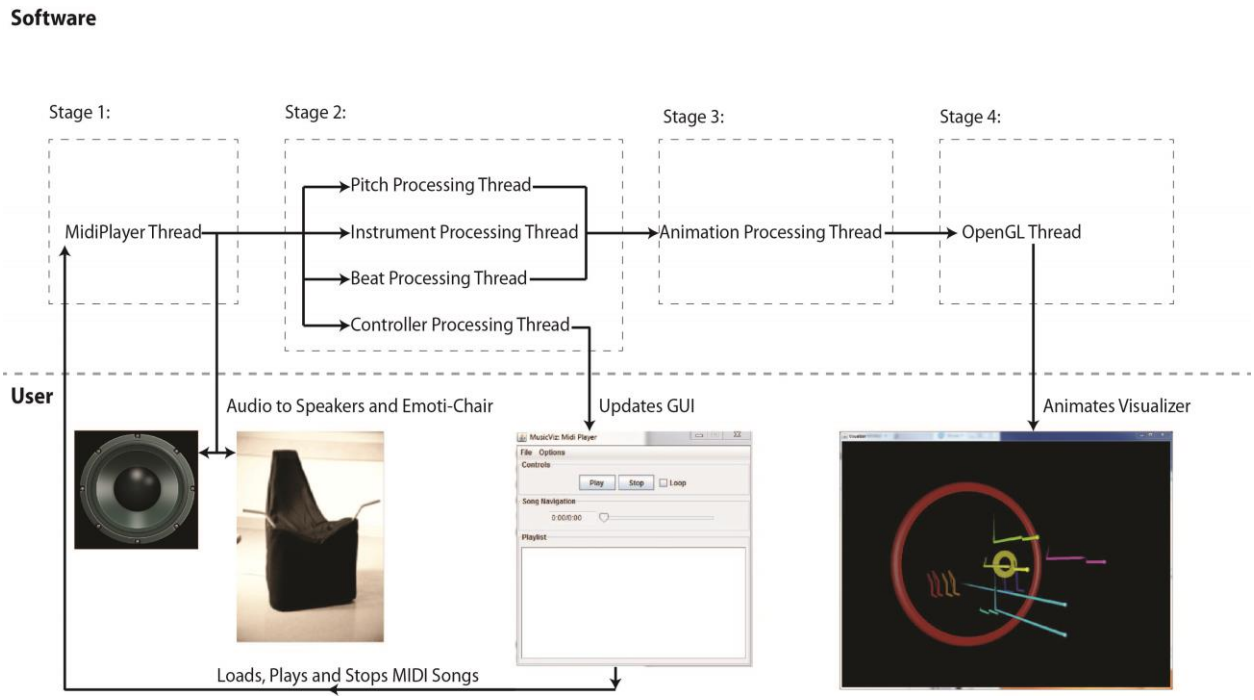


Figure 51: Four processing stages of MusicViz's visualization algorithm

The classes associated with each stage in the music animation process are described in Table 5. Each of these classes is built using Java's MIDI library and contain the code for playing, intercepting, translating and displaying and animating the music visualization.

Table 5: MusicViz's processing stages and associated classes

Stage	Class
1	MidiPlayer.java
2	PitchReceiver.java,

	BeatReceiver.java, InstrumentReceiver.java, TonalProcessor.java BeatProcessor.java
3	ConcurrentVisualizer.java ConcurrentPipe.java Beat.java
4	ConcurrentVisualizer.java ConcurrentPipe.java Beat.java

3.2.5.1 Stage 1

The class associated with the first stage is the *MidiPlayer* class. It contains controls for adjusting MIDI playback such as playing, pausing, stopping and loading MIDI files into the visualization. This class contains a single Java 6 MIDI *Sequencer* and *Synthesizer* for playback and sound generation. The *Sequencer* has four output ports (see stage 1 in Figure 51). Each output port transmits MIDI messages to each of the four *Receiver* objects in the second stage. Every *Receiver* is a processing thread as shown in Figure 51. Java code segments in Figure 79 in Appendix 6.14.1 show the *Sequencer*'s output ports connected to the *Receiver*s' input ports. Appendix 6.14.1 provides the actual code from the initialization of the *MidiPlayer* class. The user loads a MIDI song into the program by dragging a MIDI file into the GUI interface. Stage two begins after a MIDI song is loaded and the play button is triggered, which loads the MIDI sequence into the MIDI player thread.

3.2.5.2 Stage 2

The purpose of the second stage is to transform MIDI note-on/off messages into canonical form and prepare MIDI messages for the third stage. The MIDI specification does not refer to any form of a canonical MIDI message; it is solely terminology used within this thesis. As previously mentioned in

Section 3.2.1.1, MIDI note-on/off messages are not consistent. Table 4 shows that a note-off message can be created by a note-on message with a zero volume. In addition, a note-off message can contain a volume other than zero. MIDI canonicalization converts all forms of note-off variations in Table 4 to a note-off with a zero volume.

The second stage in the processing pipeline then handles the collection of MIDI messages during playback. *PitchReceiver.java*, *BeatReceiver.java* and *InstrumentReceiver.java* implement the aforementioned *Receiver* interface from the Java API. They are connected to the MIDI *Sequencer*'s output ports and receive every MIDI message during playback. Each *Receiver* processes only a specific MIDI message. *PitchReceiver.java* processes pitch-bend messages, *BeatReceiver.java* processes note-on messages for the percussion channel and *InstrumentReceiver.java* processes all note-on/off messages dedicated for non-percussion instruments. In addition, each *Receiver* object is designed to offload all processing from the Java's sound thread to a separate thread in order to improve performance. This architecture is described in Appendix 6.14.2.1.

The process of preparing MIDI messages for animation consists of calculating the affected visual construct in the music visualization by assigning a message to a specific visual construct before animation processing in the third stage. This is performed differently depending on the *Receiver* object described Section 3.2.5.2.1, Section 3.2.5.2.2 and Section 3.2.5.2.3.

3.2.5.2.1 Instrument Processing Thread

The *InstrumentReceiver.java* threads perform two functions. The first removes all note-on/off messages targeting the percussion channel and converts the message to a canonical message. The next step decides the affected visual construct. MusicViz contains a set of two pipes for each of the fifteen channels. In MusicViz, a MIDI channel has a one-to-one mapping to each pipe set. The two pipes in each channel are dedicated to displaying animations of certain notes (MIDI notes ranged from 0 to 127). A message with a pitch between 0 and 41 is animated by the channel's first pipe and a note from 42 to 127 is animated by

the channel's second pipe. The source code in for the *run* method in *InstrumentReceiver.java* is available in Appendix 6.1.

3.2.5.2.2 Beat Processing Thread

The *BeatReceiver.java* thread processes note-on messages dedicated to the percussion channel (channel 10). Beats in MusicViz are displayed as five toroid objects of varying sizes. When a percussion note is triggered, the corresponding toroid flashes and moves to the front of the animation sequence. The triggered toroid fades over time. Appendix 6.14.2.2 contains the MIDI instrument groupings syntax.

During MIDI playback, BeatReceiver.java's thread associates percussion messages with the appropriate toroid object. The resulting message is sent to the third stage (see Appendix 6.2 for the source code for the *run* method in *BeatReceiver.java*)

3.2.5.2.3 Pitch-Bend Processing Thread

The *PitchReceiver.java* thread processes pitch-bend messages, which have a status byte from 224 to 239. *PitchReceiver.java* assembles the new pitch value that affects all pitches in a specified channel by either increasing or decreasing the frequency of all notes played in the channel (see Appendix 6.12.1 for the method in which a pitch-bend message is assembled). The resulting message is sent to the third stage for animation processing (see Appendix 6.3 for the source code for the PitchReceiver class).

3.2.5.3 Stage 3

The third stage of MusicViz's processing occurs on the CPU, which is based on the *Worker Task Architecture* described by Apple [78] (See Appendix 6.14.3.1 for further information on of the *Worker Task Architecture*). The worker-thread, based on the *Worker Task Architecture*, calculates the new animations and queues them for rendering by the OpenGL thread in the fourth stage. The classes associated with the third stage are *ConcurrentVisualizer.java* (see Appendix 6.4) and *ConcurrentPipe.java* (see Appendix 6.6).

The *ConcurrentVisualizer.java* class has two associated threads in order to implement the *Worker Task Architecture* [78]. The class implements JOGL's *GLEventListener* that calls the rendering loop, which renders the OpenGL within the *display* method. The *display* method constitutes the OpenGL rendering thread. In addition, *ConcurrentVisualizer.java* extends Java's *Thread* class; the associated *run* method is overridden to perform the worker tasks on the CPU. A general architecture is shown in Figure 52. MusicViz contains sixteen animation queues, one for each MIDI channel; however, the figure shows one queue for simplicity. The animation queues are implemented using Java's *ConcurrentLinkedQueue*, which provides non-blocking and thread-safe methods for concurrent operations. The *run* method processes messages from the second stage into viable animations. The animations are queued for rendering in the fourth stage.

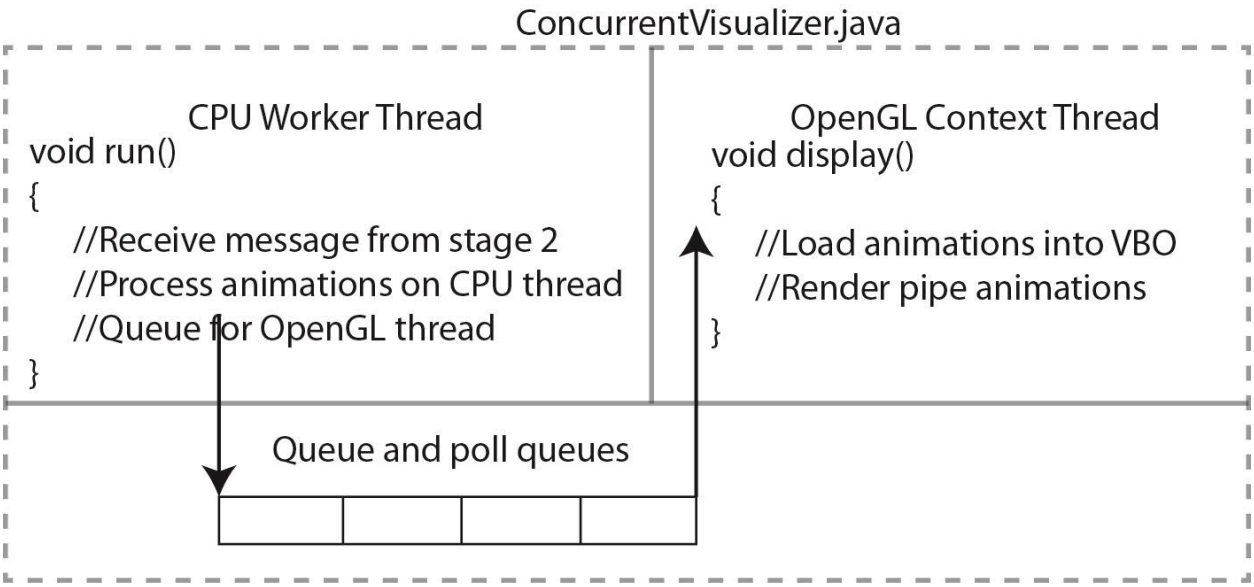


Figure 52: Pseudo-code and general structure of ConcurrentVisualizer.java's threads

3.2.5.3.1 Animating Instrument Messages

Considering OpenGL's threading model and its influence on MusicViz (see Appendix 6.14.3.1), the animation process can now be described. An instrument message (a note-on/off message for any channel but the percussion channel) contains the target channel and the specific pipe affected. The note's volume affects the pipe's size and alpha value. The note's frequency affects the pipe's y-axis value. The alpha and

size calculations are based on Equation 3 and Equation 4 respectively. The alpha value's lower-bound limit for a note-on message is 0.5 and the upper bound is 1 (see Equation 3). The lower and upper bound size is 1 and 5 respectively for a note-on (see Equation 4). The pipe's size is 0 for a note-off message. The pipe's y-axis value is mapped from the note's frequency value (see Equation 5) so that a majority of the frequencies appear in the middle of the screen, while higher notes appear at the top and lower notes at the bottom. Multiplying the frequency by a factor 5.5 provides more visual movement by increasing the movement of each pipe such that the pipe animates to the top and bottom of the user's screen. The new x, y, and z-coordinates, alpha and size values are used to calculate a new animation for the pipe. The resulting animation is queued for the OpenGL thread in the fourth stage. The corresponding code for this process is shown in the *processTones* method within the *ConcurrentVisualizer.java* class in Appendix 6.4.

Equation 3: Calculating the pipe's alpha value

$$alpha = \frac{0.5}{127} * volume + 0.5$$

Equation 4: Calculating the pipe's size

$$size = \frac{4}{127} * volume + 1$$

Equation 5: Calculating the y-axis position

$$y_{axis} = 5.5 * (frequency - 64)$$

3.2.5.3.2 Animating Pitch-Bend Messages

A pitch-bend message affects both pipes in the target channel by increasing or decreasing the pipes' y-axis positions. The message contains the offset value calculated in the second stage. The offset value is

added to each pipes' y-axis position to mimic a pitch-bend. The resulting animation is queued into the animation pipeline. This animation is rendered in the fourth stage. The corresponding code is shown in the *processPitchChanges* method in the *ConcurrentVisualizer.java* class in Appendix 6.4.

3.2.5.3.3 Animating Percussion Messages

A percussion message affects one of the five toroid objects. The affected toroid object is based on the message's status byte. The specific toroid affected is calculated in the second stage. Unlike the pitch-bend and instrument messages, this message is not queued for animation. This message immediately triggers the appropriate toroid in the OpenGL rendering thread. The triggered toroid flashes and jumps to the front of the animation sequence. As time progresses, the toroid object fades into the background. The fading occurs by changing the toroid's alpha value and is based on the distance from the front of the animation sequence (see the *draw* method within *Beat.java* in Appendix 6.5.

3.2.5.4 Stage 4

Stage four constitutes the GPU OpenGL sub-section in the *Worker Task Architecture* shown Appendix 6.14.3.1. In this stage, the position, alpha and size information are polled from each pipe's animation queues and are loaded directly into graphics memory using OpenGL VBO functions.

Pipe animation consists of percolating each face (shown in Appendix 6.14.3.1) backwards and loading the new face to the first position. Considering the pipe data is stored sequentially in VBOs, the program binds to the memory location to access the memory buffer. Animation commences by storing the first and second faces. The first face is set to the second face and the second face is pushed to the third face. This occurs until all faces are percolated backwards. Lastly, the new face is added to the first position (see *animate* method within *ConcurrentPipe.java* class for the animation code and the *draw* method within *ConcurrentVisualizer.java* for the rendering code in Appendix 6.6).

3.3 VITA: The Visually Immersive and Tactile Animation System

3.3.1 VITA Architecture

VITA is comprised of two mutually exclusive systems, MusicViz and the Emoti-Chair. Connecting MusicViz to the Emoti-Chair's Max/MSP software creates VITA, which provides a visual and tactile rendition of MIDI music. This section describes the process in which the Emoti-Chair and MusicViz are combined. Figure 53 shows an architectural overview of VITA.

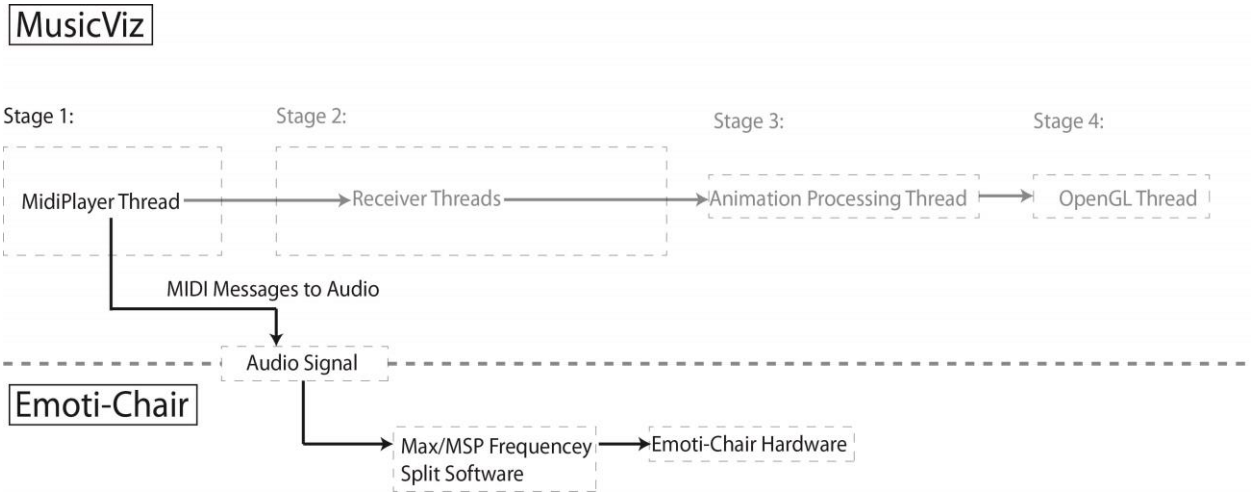


Figure 53: VITA system architecture

The Emoti-Chair is driven by software written in Max/MSP. Input is a mixed audio signal and the software then segments the signal into eight frequency bands, which range between 27.5HZ and 1,000HZ [71]. The translation algorithm is based on the frequency model described in [67].

As previously described, MusicViz uses Java's MIDI libraries for MIDI playback. The *Sequencer* reads a *Sequence* of MIDI messages from a MIDI file and sends each message to its output ports at the proper tempo and timing as described by the song. MusicViz attaches multiple *Receiver* objects to the *Sequencer* in order to intercept messages for visualization. In addition, MusicViz also attaches a MIDI *Synthesizer* to the *Sequencer*. The *Synthesizer*, which exists within the MidiPlayer Thread, generates waveform audio from the received MIDI messages. VITA is created by attaching MusicViz's audio output to the input of

the Emoti-Chair's Max/MSP frequency split software. The Max/MSP frequency software processes the incoming audio signal in real-time and presents the corresponding vibrations along the user's back.

3.3.2 VITA Playback Scenarios

The VITA system allows the user to spatially separate the various MIDI instruments between the Emoti-Chair and MusicViz using three playback scenarios. In the first scenario, percussion is played only on the Emoti-Chair and the remaining parts of the music played on MusicViz. The second scenario reverses this, where the percussion is played through MusicViz and the other instruments through the Emoti-Chair. The third scenario plays all instruments through both devices. Figure 54 shows a flow diagram of the three scenarios.

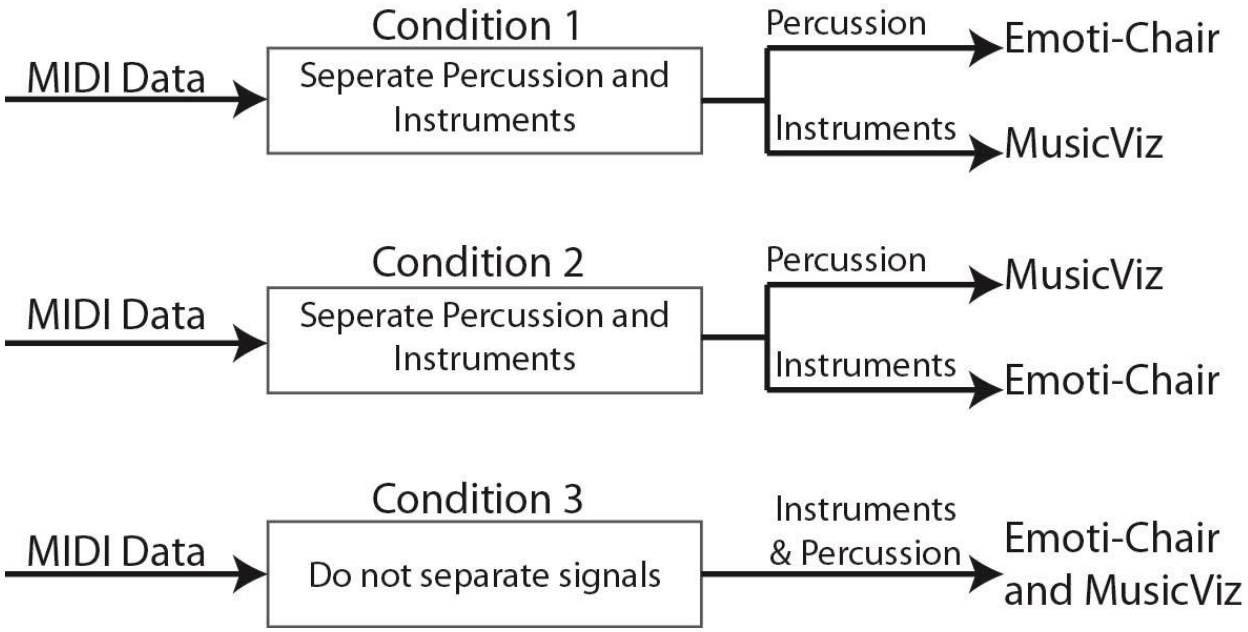


Figure 54: VITA's Spatial Separation Scenarios

3.4 Methodology

A user study is designed to evaluate the effects of the VITA system on the enjoyability, usability and effectiveness in conveying emotions to the participants (see Section **Error! Reference source not found.** for definitions). This section describes the procedure used to run the study, the research questions,

participant demographics, results and discussion. The study is approved by Ryerson's research ethics board (see Appendix 6.11).

3.4.1 Procedure

After the consent form is signed, the video camera is turned on and the participants are asked to complete a pre-questionnaire that collects demographic data such as gender, age and hearing status, as well as musical preferences and methods of accessing music.

The VITA system is then evaluated using a mixed factor design where participants listen to two blocks of six randomly selected, one-minute music samples from a variety of music genres including classical, rock, pop, country, rap/hiphop/R&B (RHRB), jazz and classical. The six musical genres were taken from iTunes's classification of genres [37]. The between-subjects component is VITA configuration conditions where each participant is randomly assigned one of three conditions and then asked to report their experience for each song. The first condition is where the instruments are played on the Emoti-chair and the drums on the visualization. The second condition is where the drums are played on the visualization and the instruments on the Emoti-Chair. The third condition is where the instruments and the drums are sent simultaneously to the Emoti-Chair and MusicViz. The within-subjects component is the six music genres. In order to reduce the possibility of an ordering effect of the data, each participant is assigned a random order for listening to each genre as well as a random song from each genre.

Pre-study, within-study and post-study questionnaires are completed for each song (see Appendix 6.8, Appendix 6.9 and Appendix 6.10). Enjoyability levels and opinions of the song are collected using a within-study questionnaire completed after each song. Participants then rate their overall experience and provide opinions of VITA once the study is complete. The study takes no more than an hour.

3.4.2 Research Questions

The main goal of the user study was to test the effectiveness of the VITA system and receive feedback from users. The following research questions were formulated to address this goal.

1. What is the impact on the experience of users from different musical genres with VITA?
2. What are the differences in the experience of VITA between H and D audiences?
3. What is the impact on users of the different VITA conditions (e.g., separating the drums from the instruments, playing the instruments on the Emoti-Chair and the drums on the visualization or vice versa, or the playing the instruments and drums simultaneously on the Emoti-Chair and MusicViz)?

3.4.3 Participants' Demographics

Twenty individuals participated in the study to determine the enjoyability and effectiveness of VITA: ten H (five male, five female) and ten D (six male, four female). An American Sign Language Interpreter participated in studies with all of the deaf participants to facilitate discussion. The participants ranged in age from 18 to 64.

Seven participants reported high school as their highest level of education (one D, six H); four participants completed college (four D); eight participants completed university (four D, four H); and one participant completed graduate school (one D). All participants reported using a computer everyday. Nine participants reported listening to music everyday (one D, eight H); five participants listen to music every two to three days (three D, two H); one participant listens to music once a week (one D); three participants listen to music once a month (three D); and two participants do not listen to music (two D).

Thirteen participants reported enjoying rock music (five D, eight H); ten participants enjoy RHRB (three D, seven H); five enjoy country music (four D, one H); four participants enjoy classical (three D, one H); three participants enjoy metal (one D, two H); and nine participants report enjoying other genres of music (seven D, two H).

Thirteen participants reported listening to music using iPods, other MP3 players and/or CDs (four D, nine H); eleven participants listen to music using various computer speaker systems (four D, seven H); three

participants listen to music through a home theater system (one D, two H); nine participants listen to music using a stereo system (five D, four H); and five participants go to live concerts (two D, three H).

Eleven participants reported listening to music for the purpose of enjoyment (three D, eight H); two participants listen to music to relax (two D); four participants listen to music for emotional experiences (one D, two H); one participant listens to music for distraction (one D); and one participant listens to music for other reasons (one D).

Three participants perform housework while listening to music (one D, two H); six participants listen to music while driving (five D, one H); four participants work while listening to music (four H); two participants exercise while listening to music; two participants do not perform any tasks while listening to music (one D, one H); and three participants perform other activities while listening to music (three D).

Ten participants use captioning or subtitles (ten D); five participants sometimes use captioning or subtitles (five H); and five participants do not use any form of subtitles or captions (five H).

3.4.4 Data Analysis

Even though the study was designed as a mixed factor design, there was insufficient power, due to the small participant numbers, to perform an analysis with all factors considered at once. Therefore, the factors were analyzed individually. First, a repeated measures ANOVA was performed with genres as the only factor (within subjects). If there were no significant differences in the dependent variables (enjoyability, emotions and effectiveness) for genres, the genre data were aggregated into a single group to allow for the data to be analyzed using the between-subjects factors (conditions and hearing status) using non-parametric analyses (Mann-Whitney U test for hearing status, or a Kruskal-Wallis H test for conditions).

In order to assess the questionnaire responses in comparison to chance and to examine correlations, the Likert scales were compressed from five-point Likert scales to three-point scales due to the lack of a

larger sample size [79]. A Spearman's rank order was performed on the unipolar emotional scales for happiness, sadness, anger and fear to examine the correlation between the scales.

4 Evaluation

4.1 Results

Even though the study was designed as a mixed factored design, there was insufficient power caused by the small number of participants to perform an analysis with all factors considered at once. Therefore, the factors were analyzed individually. The results are presented in the same order that the research questions are presented for each of the dependent variables, where the dependent variables are presented in the order in which they are laid out in the after-each song questionnaire (see Appendix 6.9). First, the results for the within-subjects variable (genres) are listed. These are followed by the results of the analysis of one of the between-subjects variable (hearing status groups) then the results of the within-subjects variable (condition). Next, the results of the genres and hearing status groups are presented and lastly, the results of the genres and conditions are presented.

4.1.1 Per-song Results

4.1.1.1 Levels of Enjoyability

4.1.1.1.1 Within Genres

To examine the differences in enjoyability between genres, a repeated measures ANOVA was carried out. There were no significant differences in the levels of enjoyability between genres ($p>0.05$). The descriptive statistics for each genre are shown in Table 29 in Appendix 6.15.2.1.

4.1.1.1.2 Within Genres Chi-Sqaure

A goodness-of-fit chi-square test was conducted to determine whether the enjoyability responses differed from chance. Participants rated their level of enjoyability on a 3-point Likert scale ranging from 1-Not enjoyable to 3-Enjoyable; one participant did not answer the question (N=19 for this analysis). There was

a significant difference for country, jazz, pop, RHRB, and rock, regardless of condition and hearing status group (see Table 6). More participants found country music enjoyable (14/20) compared to either neutral (2/20) or not enjoyable (4/20). More participants found jazz music enjoyable (13/20) compared to neutral (3/20) or not enjoyable (4/20). More participants found pop music enjoyable (17/20) compared to neutral (1/20) or not enjoyable (2/20). More participants found RHRB enjoyable (13/20) compared to neutral (4/20) or not enjoyable (2/20). Lastly, more participants found rock music enjoyable (15/19) compared to neutral (2/19) or not enjoyable (2/19).

Table 6: Descriptive statistics and p-values for chi-square values relating to the levels of enjoyability for each genre (df=2)

Genre	Mean	SD	Chi-square	P-Value
Classical	2.32	0.75	3.700	0.157
Country	2.47	0.84	12.4	0.002
Jazz	2.53	0.77	9.1	0.011
Pop	2.74	0.65	24.1	0.000
Rap	2.58	0.69	10.842	0.004
Rock	2.68	0.67	17.789	0.000

4.1.1.1.3 Between Hearing and Deaf Groups

To examine the differences in enjoyability between hearing and deaf groups, a Mann-Whitney test was carried out. There were significant differences in enjoyability between deaf and hearing individuals (U=2070, z=2.191, p=0.028). The descriptive statistics for each group are shown in Table 7.

Table 7: Descriptive statistics for the levels of enjoyability for each hearing status

Hearing Status	Mean	SD	N
Deaf	2.42	0.79	60

Hearing	2.69	0.65	58

4.1.1.1.4 Between Conditions

To examine the differences in enjoyability between conditions, a Kruskal-Wallis test was carried out. There were no significant differences in enjoyability between conditions (p=0.456). The descriptive statistics for each condition are shown in Table 30 in Appendix 6.15.1.2. See Figure 54 for conditions.

4.1.1.1.5 Within Genres and Between Hearing and Deaf Groups

To examine the differences in enjoyability within genres and between conditions, a repeated measures ANOVA was carried out. However, statistical analysis cannot be carried with both factors due to the lack of standard deviation. The descriptive statistics are shown in Appendix 6.15.1.2.

4.1.1.1.6 Within Genres and Between Conditions

To examine the differences in enjoyability within genres and between conditions, a repeated measures ANOVA was carried out. However, statistical analysis cannot be carried with both factors due to the lack of standard deviation. The descriptive statistics are shown in Appendix 6.15.1.4.

4.1.1.2 Levels of Continuous Emotions

Using the 9-point self-assessment manikin proposed by [80], participants rated the levels of valence, arousal and overbearingness representing the three dimensional model of emotion (see [2]). The first scale ranged from "1-unhappy" to "9-happy" (valence), the second scale ranged from "1-calm" to "9-excited" (arousal) and the third scale ranged from "1-subtle" to "9-overbearing" (overbearingness).

4.1.1.2.1 Within Genres

To examine the differences in conveyed emotions between genres, a repeated-measures ANOVA was carried out. The results show that there were no significant differences in the levels of valence (p=0.532), arousal (p=0.842) and overbearingness (p=0.560) between genres. Table 33 to Table 35 and Figure 84 to Figure 86 in appendix 6.15.2.1 show the descriptive statistics for each emotive factor for all genres.

A Spearman's rank-order correlation was run to assess the relationship between the rated levels of valence, arousal and overbearingness for each genre. There was a moderate correlation between arousal and overbearingness for classical music (ρ=0.533), and for country music (ρ=0.679). There was a strong correlation between arousal and overbearingness for jazz music (ρ=0.826) and pop music (ρ=0.726). There was a moderate correlation between valence and arousal for pop music (ρ=0.590). There were no correlations between the levels of valence, arousal and overbearingness for RHRB. There was a moderate correlation between arousal and overbearingness for rock music (ρ=0.604). See Table 36 to Table 41 in appendix 6.15.2.1 for all Spearman's rank-order values.

4.1.1.2.2 Between Deaf and Hearing Groups

A Mann-Whitney test was carried out to evaluate the difference in the levels of valence, arousal and overbearingness between hearing status groups. There were no significant differences in the levels of valence (p=0.535), arousal (p=0.448) and overbearingness (p=0.869) between hearing and deaf groups. Table 42 in Appendix 6.15.2.2 shows the descriptive statistics for the levels of emotions for each hearing status.

4.1.1.2.3 Between Conditions

The three conditions are as follows:

1. Percussion to the Emoti-Chair and Instruments to MusicViz
2. Percussion to the MusicViz and instruments to the Emoti-Chair
3. Percussion and Instruments simultaneously on the Emoti-Chair and MusicViz

A Kruskal-Wallis test was carried out to evaluate the difference in the levels of valence, arousal and overbearingness between conditions. The results show that there was no significant difference in the level of valence (p=0.121) or arousal (p=0.082) between conditions. However, there was a significant difference in the level of overbearingness between conditions $\chi 2(2)=6.737$, p=0.034. Pairwise comparisons were performed with a Bonferroni correction to determine the significant conditions. The

level of overbearingness was significantly different between condition two and condition three (p=0.025). Table 8 shows the descriptive statistics for the levels of emotions between each condition.

Table 8: Descriptive Statistics for the levels of emotions between conditions

Condition		Valence	Arousal	Overbearingness
1	Mean	5.57	4.93	4.97
	N	30	30	30
	SD	1.48	2.50	2.14
2	Mean	6.25	4.71	4.29
	N	24	24	24
	SD	1.89	2.53	2.07
3	Mean	6.33	5.86	5.55
	N	66	66	66
	SD	1.82	2.01	1.72

4.1.1.2.4 Within Genres and Between Hearing and Deaf Groups

A repeated-measures ANOVA was carried out to evaluate the differences in the levels of valence, arousal and overbearingness between hearing and deaf groups, and genres. There were no significant differences in the levels of valence (p=0.183), arousal (p=0.876) and overbearingness (p=0.250) between genres and hearing status. Table 43, Table 44 and Table 45 in Appendix 6.15.2.2 show the descriptive statistics for the levels of valence, arousal and overbearingness relating to each genre and hearing status.

4.1.1.2.5 Within Genres and Between Conditions

A repeated-measures ANOVA was carried out to evaluate the differences in the levels of valence, arousal and overbearingness between conditions and genres. The results show that there was no significant difference in the level of valence (p=0.557), arousal (p=0.081) and overbearingness (p=0.605) between

genres and conditions. Table 46, Table 47 and Table 48 in Appendix 6.15.2.4 show the descriptive statistics for the levels of valence, arousal and overbearingness for each genre and condition.

4.1.1.3 Levels of Discrete Emotion Ratings

Participants also rated the levels of emotions expressed in each of the visualizations using four seven-point scales for happiness, sadness, anger and fear "1" was weak and "7" was strong.

4.1.1.3.1 Within Genres

A repeated-measures ANOVA analysis was carried out in order to examine the differences in the levels of discrete emotions expressed in the songs between genres. There were no significant differences in the happiness ($p=0.700$), sadness ($p=0.720$), anger ($p=0.473$) and fear ($p=0.697$) between any of the genres. The descriptive statistics relating to the levels of discrete emotions for each genre are shown in Table 49, Table 50, Table 51 and Table 52 in Appendix 6.15.3.1.

In addition, a Spearman's rank-order correlation was run to assess the relation between the rated levels of discrete emotion for each of the genres. The correlations are shown in Table 9 to Table 14, where all non-significant correlations are grayed out. Strong correlation has a correlation coefficient above 0.7, a moderate correlation above 0.5.

Table 9: Spearman's rho correlation between the ratings of happiness, sadness, anger and fear for classical music (N=20)

Correlations between Discrete Levels of Emotions for Classical Music			Happiness Level	Sadness Level	Anger Level	Fear Level
Spearman's rho (ρ)	Happiness Level	Correlation Coefficient	1.00	0.47	0.32	0.37
		Sig. (2-tailed)		0.037	0.174	0.104

	Sadness Level	Correlation Coefficient	0.47	1.00	0.47	0.47
		Sig. (2-tailed)	0.037		0.037	0.035
	Angry Level	Correlation Coefficient	0.32	0.47	1.00	0.77
		Sig. (2-tailed)	0.174	0.037		0.000
	Fear Level	Correlation Coefficient	0.37	0.47	0.77	1.00
		Sig. (2-tailed)	0.104	0.035	0.000	

Table 10: Spearman's rho correlation between the ratings of happiness, sadness, anger and fear for country music (N=20)

Correlations between Discrete Levels of Emotions for Country Music			**Happiness Level**	**Sadness Level**	**Anger Level**	**Fear Level**
Spearman's rho (ρ)	Happiness Level	Correlation Coefficient	1.00	-0.24	-0.27	-0.28
		Sig. (2-tailed)	0.0	0.313	0.244	0.227
	Sadness Level	Correlation Coefficient	-0.24	1.00	0.66	0.74
		Sig. (2-tailed)	0.313		0.001	0.000
	Angry Level	Correlation Coefficient	-0.27	0.66	1.00	0.95
		Sig. (2-tailed)	0.244	0.001		0.000
	Fear Level	Correlation Coefficient	-0.28	0.74	0.95	1.00

		Sig. (2-tailed)	0.227	0.000	0.000	

Table 11: Spearman's rho correlation between the ratings of happiness, sadness, anger and fear for jazz music (N=20)

Correlations between Discrete Levels of Emotions for Jazz Music			**Happiness Level**	**Sadness Level**	**Anger Level**	**Fear Level**
Spearman's rho (ρ)	Happiness Level	Correlation Coefficient	1.00	-0.05	0.09	0.12
		Sig. (2-tailed)		0.851	0.709	0.617
	Sadness Level	Correlation Coefficient	-0.05	1.00	0.75	0.82
		Sig. (2-tailed)	0.851		0.000	0.000
	Angry Level	Correlation Coefficient	0.09	0.75	1.00	0.74
		Sig. (2-tailed)	0.709	0.000		0.000
	Fear Level	Correlation Coefficient	0.12	0.82	0.74	1.00
		Sig. (2-tailed)	0.617	0.000	0.000	

Table 12: Spearman's rho correlation between the discrete levels of happiness, sadness, anger and fear for pop music (N=20)

Correlations between Discrete Levels of Emotions for Pop Music			Happiness Level	Sadness Level	Anger Level	Fear Level
Spearman's rho (ρ)	Happiness Level	Correlation Coefficient	1.00	-0.02	0.24	0.32
		Sig. (2-tailed)		0.942	0.314	0.173
	Sadness Level	Correlation Coefficient	-0.02	1.00	0.49	0.75
		Sig. (2-tailed)	0.942		0.029	0.000
	Angry Level	Correlation Coefficient	0.24	0.49	1.00	0.69
		Sig. (2-tailed)	0.314	0.029		0.001
	Fear Level	Correlation Coefficient	0.32	0.75	0.69	1.00
		Sig. (2-tailed)	0.173	0.000	0.001	

Table 13: Spearman's rho correlation between the ratings of happiness, sadness, anger and fear for RHRB music (N=20)

Correlations between Discrete Levels of Emotions for RHRB Music			Happiness Level	Sadness Level	Anger Level	Fear Level
Spearman's rho (ρ)	Happiness Level	Correlation Coefficient	1.00	0.16	0.05	-0.06
		Sig. (2-tailed)		0.501	0.829	0.797

	Sadness Level	Correlation Coefficient	0.16	1.00	0.71	0.58
		Sig. (2-tailed)	0.501		0.000	0.008
	Angry Level	Correlation Coefficient	0.05	0.71	1.00	0.77
		Sig. (2-tailed)	0.829	0.000		0.000
	Fear Level	Correlation Coefficient	-0.06	0.58	0.77	1.00
		Sig. (2-tailed)	0.797	0.008	0.000	

Table 14: Spearman's rho correlation between the ratings of happiness, sadness, anger and fear for rock music (N=20)

Correlations between Discrete Levels of Emotions for Rock Music			**Happiness Level**	**Sadness Level**	**Anger Level**	**Fear Level**
Spearman's rho (ρ)	Happiness Level	Correlation Coefficient	1.00	0.15	0.14	0.17
		Sig. (2-tailed)	0.0	0.520	0.564	0.473
	Sadness Level	Correlation Coefficient	0.15	1.00	0.83	0.87
		Sig. (2-tailed)	0.520	0.0	0.000	0.000
	Angry Level	Correlation Coefficient	0.14	0.83	1.00	0.98
		Sig. (2-tailed)	0.564	0.000	0.0	0.000
	Fear Level	Correlation Coefficient	0.170	0.874	0.983	1.000

		Sig. (2-tailed)	0.473	0.000	0.000	0.0

4.1.1.3.2 Between Hearing and Deaf Groups

A Mann-Whitney test was carried out to evaluate the differences in the levels of discrete emotions expressed in the songs between hearing status groups. The test revealed that there was no significant differences in the levels of happiness (p=0.171), sadness (p=0.944) and fear (p=0.340) between hearing status. See Table 53 in Appendix 6.15.3.2 for the descriptive statistics.

4.1.1.3.3 Between Conditions

A Kruskal-Wallis test was carried out in order to examine the differences in the levels of discrete emotions expressed in the songs between conditions. The results show there were significant differences in the levels of happiness $\chi^2(2)=15.328$ (p=0.000), sadness $\chi^2(2)=16.045$ (p=0.000) and fear $\chi^2(2)=8.778$ (p=0.012).

Pairwise comparisons were performed with a Bonferroni correction to determine the significant conditions. Between condition one and three, the level of happiness, sadness and fear were significant. The level of sadness was also significant between condition two and condition three (p=0.011).

Table 15: Descriptive statistics for the levels of discrete emotions between conditions

Condition		Discrete Happiness	Discrete Sadness	Discrete Anger	Discrete Fear
1	Mean	2.67	1.07	0.93	1.20
	SD	2.45	1.57	1.20	1.73
	N	30	30	30	30
2	Mean	4.04	1.17	1.21	1.42

	SD	2.49	1.52	1.67	1.84
	N	24	24	24	24
3	Mean	4.86	2.39	2.45	2.20
	SD	1.94	1.85	1.98	1.96
	N	66	66	66	66

4.1.1.3.4 Within Genres and Between Hearing and Deaf Groups

A repeated measures ANOVA analysis was carried out to examine the differences in the levels of discrete emotions expressed in the songs between hearing status groups and genres. There were no significant differences in the happiness, sadness and fear. However, there was a significant difference in the level of anger between hearing status and genres (sphericity assumed) [F(5,90)=2.524, p=0.035]. The descriptive statistics for the discrete levels of happiness, sadness, anger and fear relating to genre and hearing status are shown Table 54, Table 55, Table 16 and

Table 56 respectively (see Appendix 6.15.3.2).

Table 16: Descriptive statistics for the levels of anger for each genre and hearing status (N=20, 10 D, 10 H)

Measure	Genre	Hearing Status	Mean	SD
Discrete Level of Anger	Classical	Deaf	1.50	1.35
		Hearing	1.50	1.27
	Country	Deaf	1.90	1.97
		Hearing	1.30	1.49
	Jazz	Deaf	2.80	2.66
		Hearing	1.60	1.27
	Pop	Deaf	2.10	2.13
		Hearing	1.20	1.23
	Rap	Deaf	1.60	2.12
		Hearing	2.10	1.85

	Rock	Deaf	1.50	2.32
		Hearing	2.80	2.35

4.1.1.3.5 Within Genres and Between Conditions

A repeated-measures ANOVA analysis was carried out in order to examine the differences in the levels of discrete emotions expressed in the songs between hearing status and genres. There were no significant differences in the discrete levels of happiness (p=0.727), sadness (p=0.256), anger (p=0.709) and fear (p=0.268) between the conditions and genres. The descriptive statistics for the discrete levels of happiness, sadness, anger and fear relating to genre and conditions are shown Table 57, Table 58, Table 59 and Table 60 respectively in Appendix 6.15.3.4.

4.1.1.4 Levels of Distraction and Focus

Participants were asked to rate their level of distraction from the visualization and Emoti-Chair on two three-point scales. The first scale ranged from "1-I was very distracted from the visualization" to "3-I was not distracted from the visualization". The second scale ranged from"1-I was very distracted from the Emoti-Chair" to "3-I was not distracted from the Emoti-Chair". In addition, the participants were asked to rate their focus on the entire system where 1 was "I was focused on the visualization and did not focus on the Emoti-Chair", 2 was and "3-I focused only on the Emoti-Chair and did not focus on the visualization". These scales are not ordinal and therefore repeated measures ANOVAs cannot be conducted on the data. Thus only a chi-sqaure for association is conducted.

4.1.1.4.1 Within Genres Chi-Square

A goodness-of-fit chi-square test was conducted on the levels of distraction from the visualization, Emoti-Chair and on the entire system as a whole. For levels of distraction from the visualization, the chi-square test was statistically significant for jazz, pop, RHRB and rock (see Table 17). More participants were focused on the visualization (13 for jazz, 13 for pop, 11 RHRB, 11 for rock) compared to participants that

did not notice their level of distraction (1 for jazz, 2 for pop, 2 for RHRB, 1 for rock) or participants that were distracted (6 for jazz, 4 for pop, 7 RHRB, 8 for rock).

Table 17: Chi-Square results relating to the levels of distraction from the visualization for each genre (df=2)

Genre	Chi-square	Mean	SD	P-Value	N
Classical	2.00	2.16	0.90	0.368	19
Country	2.63	2.00	0.94	0.268	19
Jazz	10.90	2.35	0.93	0.004	20
Pop	10.84	2.47	0.84	0.004	19
RHRB	6.10	2.20	0.95	0.047	20
Rock	7.90	2.15	0.10	0.019	20

For the levels of distraction from the Emoti-Chair, a chi-square test was statistically significant for all six genres (see Table 18). More participants were focused on the Emoti-Chair (12 for classical, 15 for country, 13 jazz, 14 for pop, 13 for RHRB, 16 for rock) compared to participants that did not notice their level of distracted (5 for classical, 1 for country, 2 jazz, 2 for pop, 2 for RHRB, 1 for rock) or participants that were distracted (3 for classical, 4 for country, 5 jazz, 4 for pop, 5 for RHRB, 3 for rock).

Table 18: Chi-Square results relating to the levels of distraction from the Emoti-Chair for each genre (dF=2 and N=20)

Genre	Chi-square	Mean	SD	P-Value
Classical	6.70	2.45	0.76	0.035
Country	16.30	2.55	0.83	0.000
Jazz	9.70	2.40	0.88	0.008
Pop	12.40	2.50	0.83	0.002

Rap	9.70	2.40	0.88	0.008
Rock	19.90	2.65	0.75	0.000

For the levels of focus on the whole system, the chi-square test was statistically significant for country and RHRB (see Table 19). More participants focused only on the Emoti-Chair and ignored the visualization (11 for country, 9 for RHRB) compared to participants that focused equally on the Emoti-Chair and MusicViz (7 for country, 10 for RHRB) or participants that focused only on the visualization and did not focus on the Emoti-Chair (2 for country, 3 for RHRB).

Table 19: Chi-Square values results to the levels of focus for each genre

Genre	Chi-square	Mean	SD	P-Value	N
Classical	1.68	2.21	0.86	0.431	19
Country	6.10	2.45	0.69	0.047	20
Jazz	1.90	2.15	0.75	0.387	20
Pop	1.90	2.15	0.75	0.387	20
Rap	7.30	2.40	0.60	0.026	20
Rock	3.70	2.20	0.70	0.157	20

4.1.1.5 Level of Agreement

Participants were asked to rate their level of agreement with each of the following seven statements on a 3-point Likert-Scale ranging from "1-Agree" to "3-Disagree". The seven statements were as follows. 'Understand' was considered as a participant's ability to experience emotions from the music.

1. The movement was hard to watch
2. The patterns and shapes on the screen were pleasant to watch
3. The vibrations helped me understand the visualization

4. The visualization helped me understand the vibrations
5. The vibrations were too strong
6. The vibrations up and down my back felt pleasant
7. The pattern of vibrations on my back was confusing

4.1.1.5.1 Within Genres

A repeated measure ANOVA was used to determine the differences in the levels of agreement with the each of the seven statements shown above between genres. Sphericity was not met and therefore a Hyund-Feldt (epsilon=0.831) was used. There were no differences in the levels of agreement between genres for statements 1 (p=0.912), 2 (p=0.655), 3 (p=0.542), 4 (p=0.829), 6 (p=0.942), and 7 (p=0.840). However, there was a significant difference in the level of agreement between genres for the statement "the vibrations were too strong" $F(4.155,66.477)=2.561$, $p=0.044$.

Paired t-tests, using a Bonferroni correction, showed that there was a significant difference between classical and country (p=0.034), country and jazz (p=0.030), and country and pop (p=0.008).

The descriptive statistics are shown in Appendix 6.15.4.1 for Table 61, Table 62, Table 63, Table 64, Table 65, Table 66 and below for Table 20 for each of the seven statements shown above.

Table 20: Descriptive statistics for the levels of agreement with statement 5 for each genre (N=17)

Measure	Genre	Mean	SD
The vibrations were too strong	Classical	2.76	0.56
	Country	2.24	0.90
	Jazz	2.65	0.49
	Pop	2.76	0.44
	Rap	2.41	0.87
	Rock	2.59	0.71

4.1.1.5.2 Between Hearing and Deaf Groups

A Mann-Whitney test was carried out to evaluate the differences in the levels of agreement between hearing status for all statements. There were no significant differences in the levels of agreement with any of the seven statements above. P-values for each of the statements are as follows: (1) p=0.530; (2) p=0.209; (3) p=0.633; (4) p=0.284; (6) p=0.176; and (7) p=0.547. The descriptive statistics are shown in Table 67 and Table 68 in Appendix 6.15.4.2.

4.1.1.5.3 Between Conditions

A Kruskal-Wallis test was carried out in order to examine the differences in the levels of agreement with the each of the seven statements shown above between each condition and genre. The Kruskal-Wallis test revealed that statements 1 to 4 were statistically significant between conditions as follows: (1) $X^2(2)=8.452$ p=0.015; (2) $X^2(2)=8.626$ p=0.013; (3) $X^2(2)=10.896$ p=0.004; (4) $X^2(2)=12.852$ p=0.002. Pairwise comparisons were performed with a Bonferroni correction for multiple comparisons for each of the statistically significant statements. Between conditions one and three, participant's levels of agreement were significant for statements 1, 2, 3 and 4. The descriptive statistics are shown below in Table 21 and Table 69.

Table 21: Descriptive statistics for the levels of agreement for statements 1 to 4 for each condition

Condition		The Movement on the screen was hard to watch	The patterns and shapes on the screen were pleasant to watch	The vibrations helped me understand the visualization	The visualization helped me understand the vibrations
1	Mean	2.03	1.97	2.14	2.21
	SD	0.10	0.93	0.97	0.96
	N	30	30	28	28
2	Mean	2.57	1.38	1.57	1.57
	SD	0.75	0.70	0.81	0.75
	N	21	21	21	21
3	Mean	2.61	1.47	1.47	1.48
	SD	0.74	0.81	0.81	0.83
	N	66	66	66	66

4.1.1.5.4 Within Genres and between Hearing and Deaf Groups

A repeated measure ANOVA was used to determine the differences in the levels of agreement with the each of the seven statements between hearing status and genre. There were no significant differences in the levels of agreement with any of the above seven statements between deaf and hearing participants. The p-values for each of the seven statements are: (1) p=0.176; (2) p=0.281; (3) p=0.767; (4) p=0.546; (5)

p=0.573; (6) p=0.758; and (7) p=0.473. The descriptive statistics are shown in Table 70, to Table 76 in Appendix 6.15.4.2.

4.1.1.5.5 Within Genres and Between Conditions

A repeated measures ANOVA was used to determine the differences in the levels of agreement within genres and between conditions for each of the seven statements previously shown. However, statistical analysis cannot be carried with both factors due to the lack of variance. The descriptive statistics are shown in Appendix 6.15.4.5.

4.1.2 Post-study Results

After each participant completed the within-study questionnaire, they were asked to provide additional feedback pertaining to their overall experience with the VITA system in the post-study questionnaire. The feedback pertained to the overall level of enjoyability of VITA, participants' ideas as to what the visual constructs in MusicViz meant, and their level of agreement with the four following statements, NASA TLX and open-ended written feedback. See Appendix 6.10 for the post-study questionnaire.

4.1.2.1 Levels of Enjoyability

4.1.2.1.1 Chi-Square

A chi-square goodness-of-fit test was conducted on the enjoyability ratings. The chi-square test was statistically significant for enjoyability $\chi^2(2)=28.90$, $p = 0.00$ ($M= 2.85$, $SD = 0.49$) More participants found the VITA system "Enjoyable" (18/20) compared to individuals who found the system "Not Enjoyable" (1/20) and "Neutral" (1/20).

4.1.2.1.2 Between Hearing and Deaf Groups

A Mann-Whitney test was carried out to evaluate the differences in enjoyability between hearing status groups. There were no significant differences in enjoyability between each hearing status (p=0.481). The descriptive statistics are shown below in Table 85.

4.1.2.1.3 Between Conditions

A Kruskal-Wallis test was carried out in order to examine the differences in enjoyability between conditions. There were no significant differences in enjoyability between conditions ($p=0.423$). The descriptive statistics are shown below in Table 84.

4.1.2.2 Meaning of the Visual Constructs in the Visualization

One set of questions pertained to what the participants thought the visual constructs in MusicViz meant. The participants were asked to select: (1) intensity; (2) frequency; (3) beat; (4) grouping; or (5) other. The questions are as follows:

1. What does the brightness in the visualization mean to you?
2. What do the shapes in the visualization mean to you?
3. What do the different heights of the pipes on the screen mean to you?

4.1.2.2.1 Between Hearing and Deaf Groups

A frequency analysis was used to examine how participants' answers varied based on their hearing status for the third question. The results are shown in Figure 55, Figure 56, Figure 57.

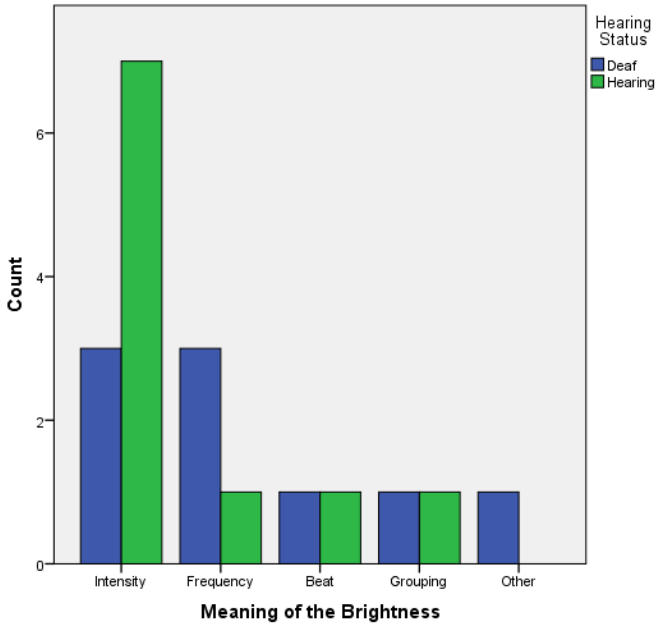


Figure 55: Frequency chart for participants' answers for each hearing status pertaining to the meaning of brightness in the visualization

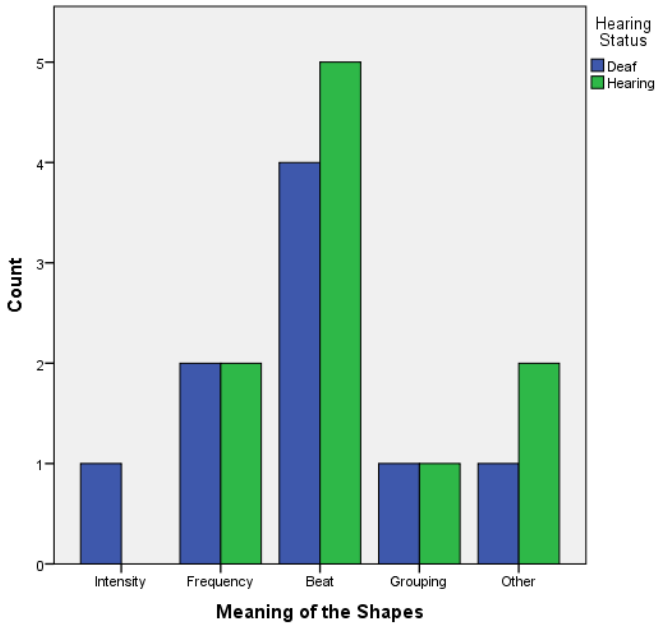


Figure 56: Frequency chart for participants' answers for each hearing status pertaining to the meaning of the shapes in the visualization

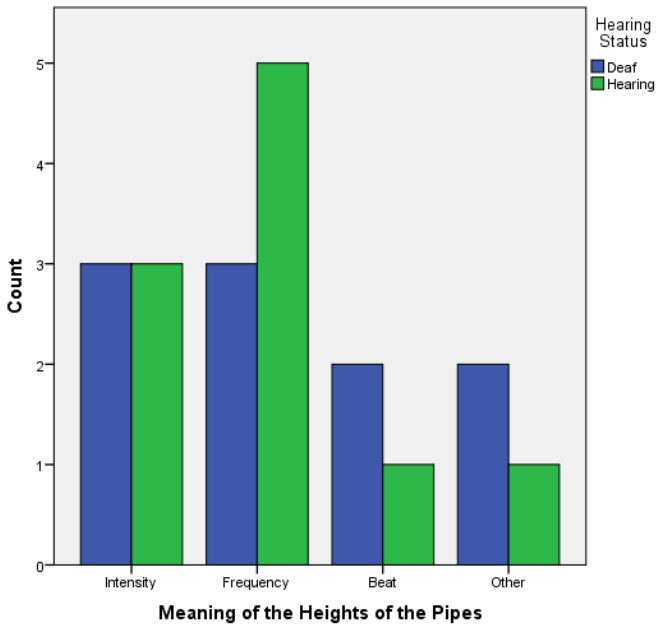


Figure 57: Frequency chart for participants' answers for each hearing status pertaining to the meaning of the heights of the pipes in the visualization

4.1.2.2.2 Between Conditions

A frequency analysis was used to see how participants' answers varied based on the assigned condition for the three questions stated above. The results are shown in Figure 58, Figure 59 and Figure 60.

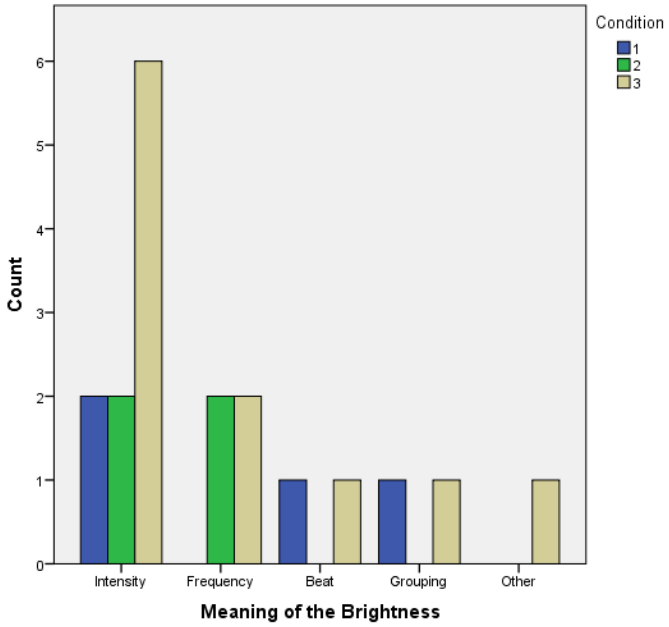


Figure 58: Frequency chart for participants' answers for each condition pertaining to the meaning of brightness in the visualization

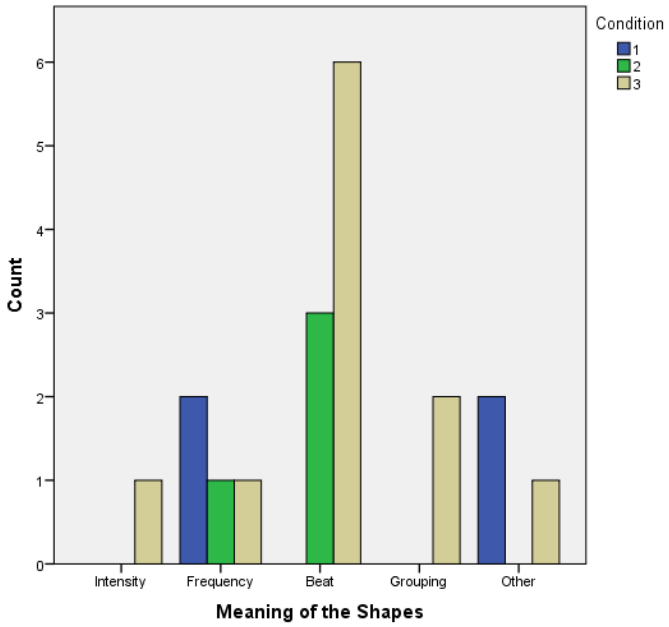


Figure 59: Frequency chart for participants' answers for each condition pertaining to the meaning of the shapes in the visualization

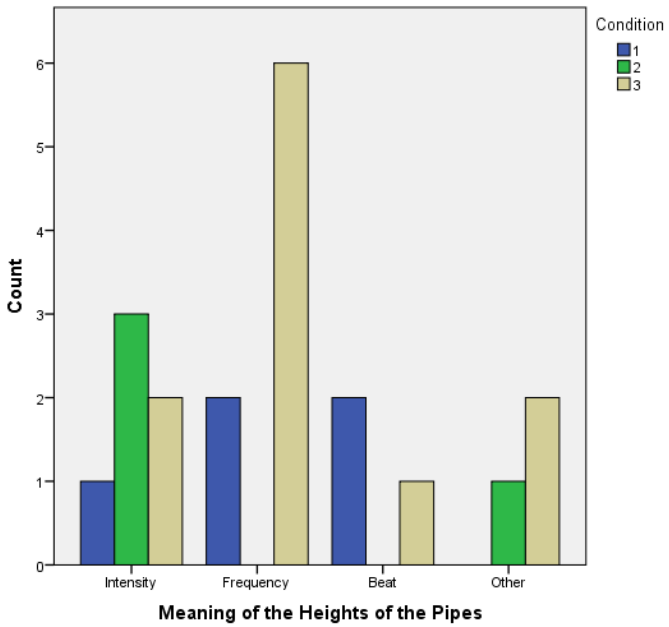


Figure 60: Frequency chart for participants' answers for each condition pertaining to the meaning of the heights of the pipes in the visualization

4.1.2.3 Levels of Agreement

Participants were asked to rate their level of agreement on a 3-point Likert scale where "1-Agree" and "3-Disagree" with the following four statements. The statements will be referred to by their number in the list below. 'Understand' was considered as a participant's ability to experience emotions from the system.

1. The vibrations helped me understand the visualization
2. The visualization helped me understand the vibrations
3. I was distracted throughout the experience
4. The experience was comfortable

4.1.2.3.1 Chi-Square

A goodness-of-fit chi-square test was conducted on the level of agreement for each of the above four statements to compare the responses with chance. All statements had a significant chi-square result (see

Table 22). More participants agreed that the vibrations helped them experience the visualization (18/20) compared to participants that disagreed (2/20). More participants agreed that the visualization helped them experience the vibrations (13/20) compared to individuals that did not care (2/20) or to individuals that disagreed (5/20). More participants were not distracted throughout the experience (16/20) compared to individuals who were distracted (4/20). Lastly, more individuals agreed that the experience was comfortable (17/20) compared to individuals that did not care (2/20) and to individuals that though the experience was not comfortable (1/20).

Table 22: Descriptive statistics for the levels of agreement of each statement

	Statement 1	Statement 2	Statement 3	Statement 4
Mean	1.20	1.60	2.60	1.20
N	20	20	20	20
SD	0.62	0.88	0.82	0.52
Chi-Square	12.80	9.70	7.20	24.10
P-value	0.000	0.008	0.007	0.000
Df	1	2	1	2

4.1.2.3.2 Between Hearing and Deaf Groups

A Mann-Whitney test was carried out to evaluate the differences in the levels of agreement between hearing status groups for the four statements. There were no significant differences in the levels of agreement for any of the four statements. The p-values for each statement are as follows: (1) p=0.481; (2) p=0.684; (3) p=0.481; and (4) p=0.684. Table 86 shows the descriptive statistics.

4.1.2.3.3 Between Conditions

A Kruskal-Wallis analysis could not be performed due to the lack of standard deviation in the answers provided.

4.1.2.4 NASA Task-Load Index

As a section of the post-study questionnaire, each participant was asked to assess their level of workload on NASA's Task-Load Index [81], which uses multi-dimensional rating scales. The dimensions used in the NASA-TLX are: (1) mental demand; (2) physical demand; (3) temporal demand; (4) performance; (5) effort; and (6) frustration.

The NASA TLX evaluation consists of two parts. The first part is for participants to provide a rating for each dimension. The ratings are selected in intervals of five from 0 to 100. The second part asked participants to provide a weighting for each of the dimensions by selecting the dimension that most contributed to workload among 15 pairs. How often one dimension is selected determines its importance (one a scale of 0 to 5). These weights are then are multiplied against each of the dimensions. The results are a normalized rating across each participant for each dimension that is comparable. After the weight is applied, the rating scale ranges from 0 to 500.

4.1.2.4.1 Mental Demand

The "Mental Demand" rating scale ranges from "Low" to "High" and pertains to the amount of mental and perceptual activity that the participant thought they required to experience the VITA system. The mean and standard deviation for mental demand for all participants is (M=184.74, SD=128.38, N=19). Five participants scored their mental demand between 0 and 100. Five participants rated their score between 100 and 200. Six participants rated their mental demand between 200 and 300. One participant rated their score between 300 and 400. Two participants rated their score between 400 and 500.

4.1.2.4.2 Physical Demand

The "Physical Demand" rating scale ranges from "Low" to "High" and pertains to the amount of physical activity that the participant thought they required to experience the VITA system (M=33.74, SD=63.33, N=19). Fifteen participants rated their level of physical demand between 0 and 50. 2 participants rated

their levels between 50 and 100. One participant rated their level between 100 and 200. Last, one participant rated their level between 200 and 250.

4.1.2.4.3 Temporal Demand

The "Temporal Demand" rating scale ranges from "Low" to "High" and pertains to the amount of time pressure that the participant felt when carrying out the task (M=84.95, SD=97.10, N=19). Ten participants rated their level of temporal demand between 0 and 50. Three participants rated their score between 50 and 100. One participant rated their score between 100 and 150. One participant rated their score between 150 and 200. Three participants rated their score between 200 and 250. One participant rated their score between 300 and 350.

4.1.2.4.4 Performance

The "Performance" rating scale ranges from "Perfect" to "Failure" and asks the participants to rate their successfulness in accomplishing the goals of the task set before them (M=82.37, SD=100.39, N=19). Nine participants rated their score between 0 and 50. Two participants rated their score between 50 and 100. Five participants rated their score between 100 and 150. Two participants rated their score between 150 and 200. One participant rated their score between 400 and 450.

4.1.2.4.5 Effort

The "Effort" rating scale ranges from "Very Low" to "Very High" and pertains to the amount of effort (physical and mental effort) to accomplish their level of performance (M=125.53, SD=75.44, N=19). Four participants rated their score between 0 and 50. Two participants rated their score between 50 and 100. Seven participants rated their score between 100 and 150. Four participants rated their score between 150 and 200. One participant rated their score between 250 and 300. One participant rated their score between 300 and 350.

4.1.2.4.6 Frustration

Lastly, the "Frustration" rating scale ranges from "Low" to "High" and pertains to the participant's level of insecurity, discouragement, irritation and stress while carrying out the task (M=70.00, SD=104.24, N=19). Thirteen participants rated their score between 0 and 50. Three participants rated their score between 100 and 150. One participant rated their score between 150 and 200. One participant rated their score between 200 and 250. Two participants rated their score between 250 and 300.

4.2 Discussion

This section describes the user study findings in relation to the research questions. As a reminder, the research questions are listed below. The discussion answers the

1. What is the impact on the experience of users from different musical genres with VITA?
2. What are the differences in the experience of VITA between H and D audiences?
3. What is the impact on users of the different VITA conditions (e.g., separating the drums from the instruments, playing the instruments on the Emoti-Chair and the drums on the visualization or vice versa, or the playing the instruments and drums simultaneously on the Emoti-Chair and MusicViz)?

4.2.1 Per-song Discussion

4.2.1.1 Levels of Enjoyability

4.2.1.1.1 Within Genres

Examining the results for genre, it appears that a majority of the participants rate the experience of country, jazz, pop, rock and RHRB, regardless of condition and hearing status, as most enjoyable, where 17/20 (85%) rated pop enjoyable, 15/19 (79%) for rock, 14/20 (70%) for country, 13/20 (65%) jazz, and 13/19 (68%) for RHRB. The highest rating of enjoyability is for pop (M=2.75, SD=0.64), followed by rock (M=2.68, SD=0.67) and RHRB (M=2.58, SD=0.96). [10] found that rock music is the most enjoyable when viewing only the visualization component of VITA. Similarly, [10] found that, even

though RHRB has a high rating, its ratings are the lowest of the group when viewing only the visualization.

This may be because pop, rock and RHRB have simpler visualizations in that there are fewer instruments on the screen as well as fewer instruments on the vibro-tactile chair. Fewer instruments played through the vibro-tactile chair may make each vibration more comprehensible. In addition, fewer instruments on the visualization component and vibro-tactile component may create a more enjoyabile experience for emotional interpretation.

It was hypothesized that RHRB music would receive the highest enjoyability rating especially among the D participants as it has a strong bass and music with a strong bass is preferred by D people; this likely because bass frequencies can be felt by the body. Similarly, it was hypothesized that country and jazz music would have low enjoyability ratings because they lack a strong base. However, a majority of participants found them enjoyable, where 14/20 (70%) of participants enjoyed country (M=2.5, SD=0.83) and 65% rated jazz as enjoyable (M=2.45, SD=0.83). These results are similar to a study carried out by [10], where it was found that, although RHRB was enjoyable, it was the least enjoyable of all genres. It would appear, however, that when music visualization and vibration are combined, more genres of music are preferred and that RHRB is one of the top three enjoyable genres.

Classical music was the only genre that seemed to be less enjoyable (M=2.35, SD=0.75) and there was no difference from chance for the enjoyability rating. Classical music is performed by orchestras containing more instruments than any other genre. As result, MusicViz must make use of all 45 pipes and the 5 percussion visuals producing a complex visualization on screen that is perhaps difficult to perceive. Classical music is also inherently more complex musically with the changes of pitches and volume between multiple categories of instruments. This translates into not only a fuller visualization but also a highly complex visualization with much movement that may saturate the visualization and the Emoti-Chair. This complexity may be confusing to audiences and thus less enjoyable. These results indicate that

it may be useful to allow users to select the number of pipes or vibro-tacile stimulators to activate, when playing music through VITA. Various visualization techniques such as the *focus+context* idea may be used to allow users to dim pipes into the background in order to focus on the instruments that are important to their experience. In the future, the users should be given the options to increase or decrease the amplitude of certain actuators to maximize their experience and minimize confusing.

4.2.1.1.2 Between Deaf and Hearing Groups

Examining the results for D and H groups, it appears that, in general, H participants enjoy the VITA system more (M=2.69, SD=0.65) than the D participants (M=2.42, SD=0.79). This was surprising because it was hypothesized that D participant would give VITA a higher enjoyability rating due to the similarity in how D individuals experience music. D communities experience music through a strong bass presence that is felt by being in close proximity to the sound source, which is similar to the Emoti-Chair. Throughout the study, D participants frequently offered their own experiences of how they experience music. Many participants indicated that at D parties, the speakers were laid flat on the ground and turned to the maximum volume in order to feel the vibrations through the entire floor and walls. The higher enjoyability rating givento VITA by H participants may be attributed to the fact that experiencing music through vibrations, while artificially deaf, is a new experience that acted as a catalyst for a more enjoyable and unique experience. H participants "liked the correlation between the vibrations and visual aspects", "liked being able to somewhat map the vibrations to the visualizations and 'feel' them", another felt "immersed in the song" and another participant stated that "the visualization … conveyed meaning". Even though H participants appeared to enjoy the VITA system more than D participants, the chi-square results for the post-study questionnaires appear to show that the experience was an overall enjoyable experience regardless of the hearing status (M=2.85, SD=0.49) as well as the participants appeared to agree that the overall experience was comfortable (M=1.20, SD=0.52). It is promising that D and H participants appeared to enjoy VITA even though D participants enjoyed the experience less. However, this indicates that the enjoyability experience may be improved for D participants. Unlike H participants,

D participants communicated suggestions that may improve VITA. Participants stated that they have been let down by previous visualization and prefer to rely on being in close proximity to loud speakers. D individuals' experience may benefit from having greater control over the intensity of the vibro-tactile music and greater control over the visualization's pipes and percussion components. Greater control may provide D individuals with a more customizable and unique experience.

4.2.1.2 Levels of Discrete and Continuous Emotions

4.2.1.2.1 Within Genres

Examining the details of the emotional ratings, there is a positive correlation between the level of arousal and level of overbearingness for classical, country, jazz and rock music (see Table 36 to Table 41). The correlation between arousal and overbearingness seem to be a logical link because as a musical piece becomes more exciting or faster/increased pitch changes, the pipes will move faster and/or increase the y-axis changes, which may translate into more visual-vibratory complexity. More visual complexity may make the music more difficult to track and thus make the music more overbearing. [2] report that pitch, volume and tempo can interact and affect the other to create more complex expressions of emotions and associations. This may happen in VITA and as a result, the visual and vibro-tactile displays become more complicated.

Pop music had a correlation between valence and arousal. This correlation between valence and arousal has also been reported for auditory music. [4] reports that music with higher levels of enjoyment have correlation between valence and arousal. This may provide an explanation as to why pop music is the most enjoyable genre.

For the measures of discrete unipolar emotions, there is a positive correlation between sadness and anger; sadness and fear; and anger and fear for country (Figure 62), jazz (Figure 63), RHRB (Figure 65) and rock (Figure 66) music. Participants appeared to recognize that a song is as angry and fearful as it is sad. These correlations may be supported by the literature, where the authors of [82] present research showing

that anger and sadness were intimately linked to each other. Participants' answers also show a positive correlation between angry and fear for classical music (see Figure 61). Also participant's answers show a positive correlation between sadness and fear as well as between angry and fear for pop music (see Figure 64). Even though there was a low correlation between sadness and anger for classical and pop music and a low correlation between sadness and fear for classical music, the trends do seem to follow the results found in [82]. It is promising that participants, regardless of hearing status, appear to recognize emotions..

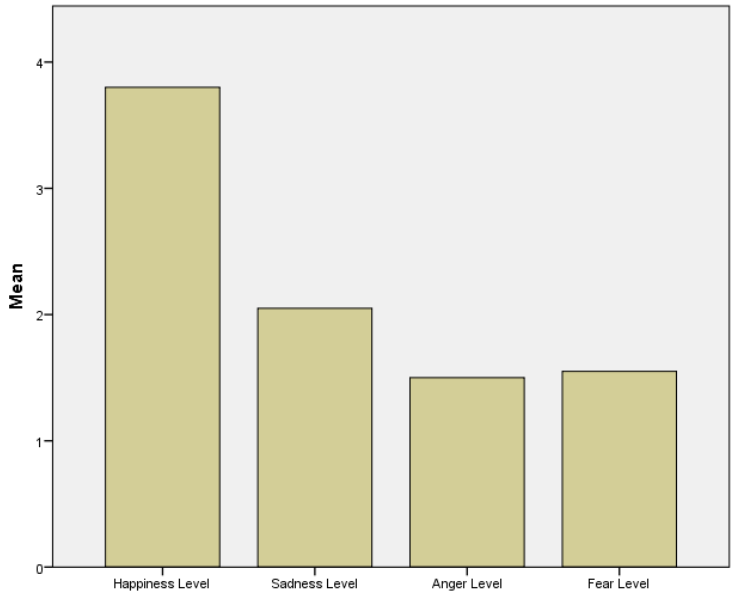


Figure 61: Descriptive statistics for the levels of discrete emotions for classical music (N=20)

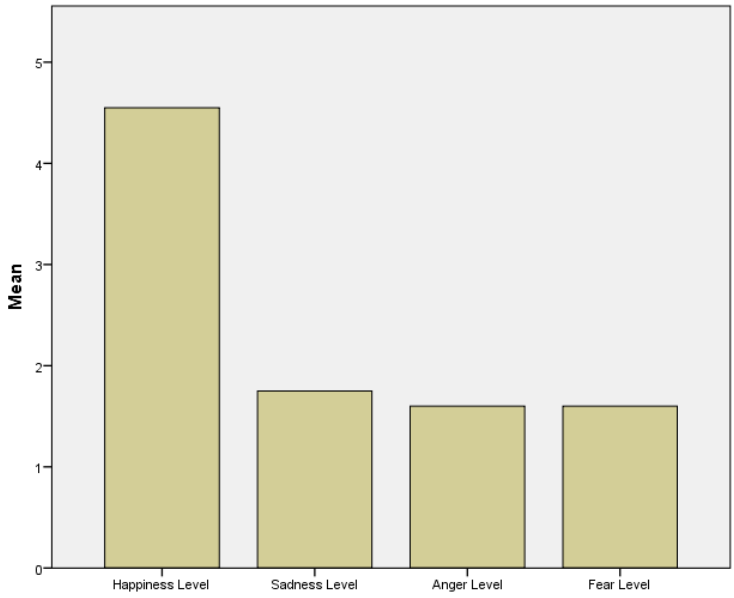


Figure 62: Descriptive statistics for the levels of discrete emotions for country music (N=20)

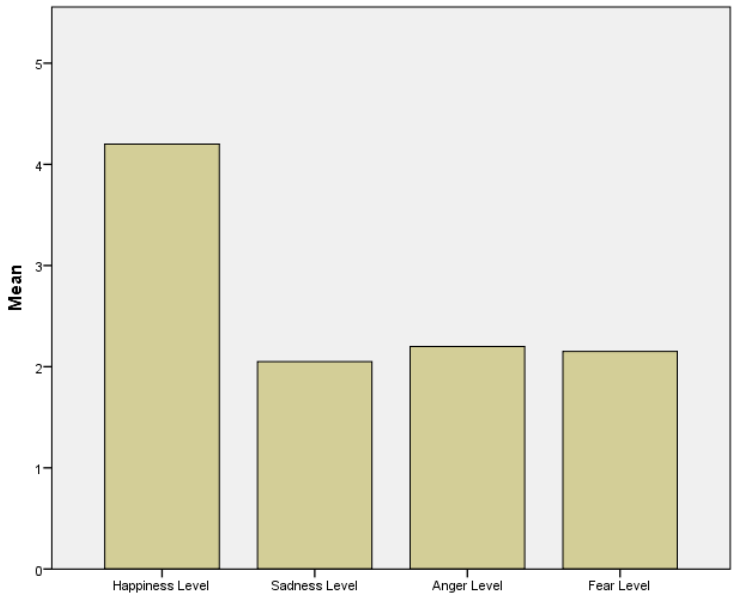


Figure 63: Descriptive statistics for the levels of discrete emotions for jazz music (N=20)

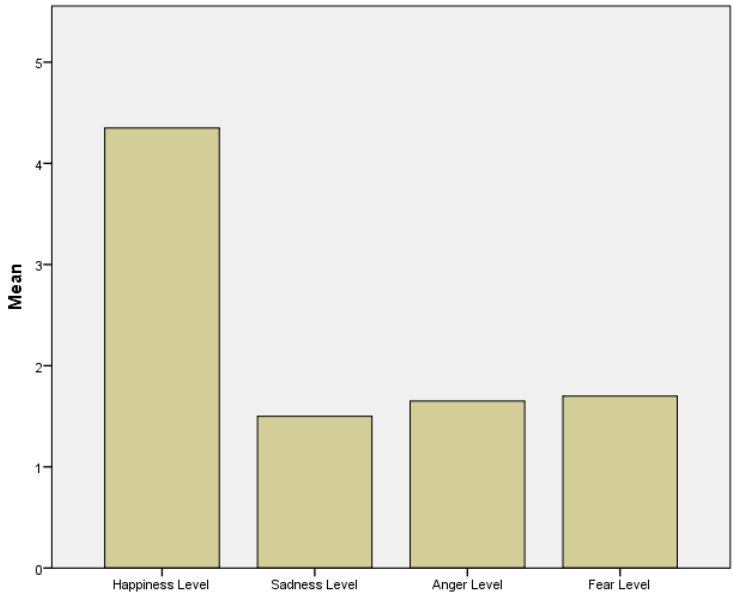


Figure 64: Descriptive statistics for the levels of discrete emotions for pop music (N=20)

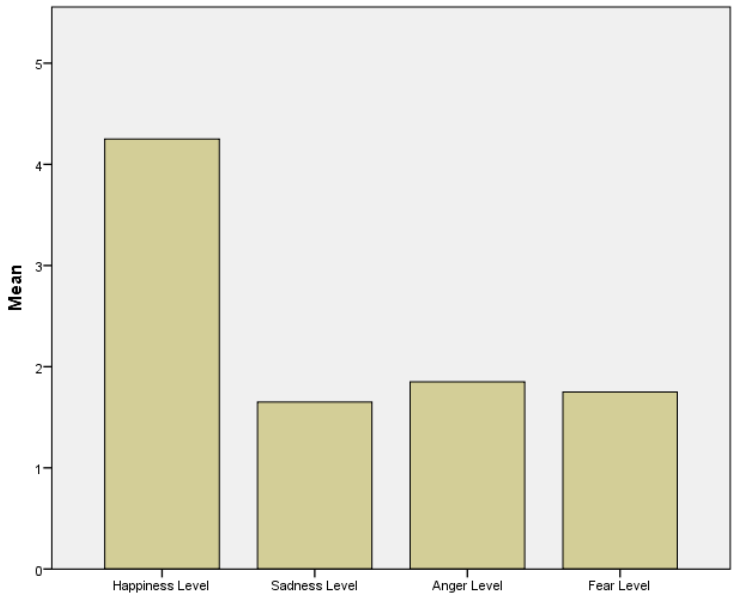


Figure 65: Descriptive statistics for the levels of discrete emotions for RHRB music (N=20)

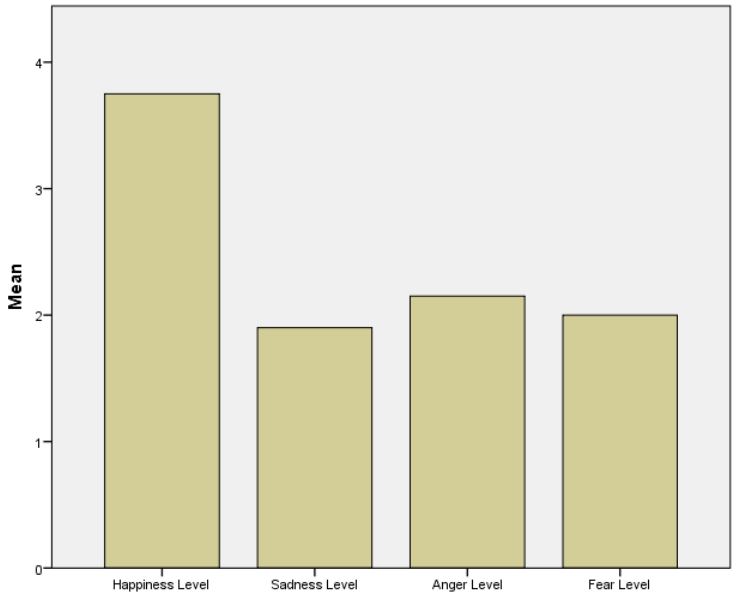


Figure 66: Descriptive statistics for the levels of discrete emotions for rock music (N=20)

4.2.1.2.2 Between Conditions

As a reminder, the conditions are as follows:

4. Percussion to the Emoti-Chair and Instruments to MusicViz
5. Percussion to the MusicViz and instruments to the Emoti-Chair
6. Percussion and Instruments simultaneously on the Emoti-Chair and MusicViz

Examining the results of the overbearingness, participants appear to recognize that condition three (M=5.15, SD=1.72) is more overbearing than condition two (M=4.29, SD=2.07). This can be attributed to the characteristics of each condition. The second condition splits percussion and instruments into two separate signals each with a smaller amount of instruments compared to the signal in the third condition. The third condition contains one signal representing the entire percussion and instruments and it may be more overbearing because it conveys more information through each device than condition two. However, [6] shows that sensory redundancy is positive because it increases robustness of learning and can reinforce conveyed concepts.

Four participants who experienced the third condition on VITA stated that "it's interesting when the vibrations match the beats and movement in the visualization", another "liked being able to somewhat map the vibrations to the visualization and feel them", another "liked the correlation between the vibrations and visual aspect", lastly another stated that VITA "allows them to see/notice the changes better and pick up on subtle sounds better via visual + vibration matching". It was a common comment throughout the different conditions that participants attempted to use the visualization and Emoti-chair as a positive feedback method of learning how each of the devices worked. Even though participants found condition three more overbearing, it does not mean that the participants thought their performance was worse. As shown in Figure 67 and Table 23, participants performing condition three (M=84.00, SD=57.53) thought they achieved better performance than the participants performing condition two (M=146.25, SD=198.05). In addition, participants performing condition three thought they used less mental demand (M=176.50, SD=143.27) versus individuals in condition two (M=217.50, SD=176.50). This is surprising because it was originally hypothesized that to achieve better performance, participants would have to use a higher mental demand. However, participants performing condition three appeared to use an overall greater effort (M=127.50, SD=83.208) than participants performing condition two (M=82.50, SD=45.74) to accomplish their higher level of performance. This is logical considering the information was more overbearing.

The results indicate that condition three may be the best condition with which to experience VITA. Condition three -although more overbearing- provides a consistent signal to both the visual and vibro-tactile domain such that participants can redundantly experience the same signal in two different domains. These participants found they were more successful in using VITA which may have been as a result of the sensory redundancy provided in condition two. This may allow the participants to think less to experience the signal. This may also make VITA easier to use for experiencing music. This is important to composers and individuals who wish to experience music through or compose music for VITA.

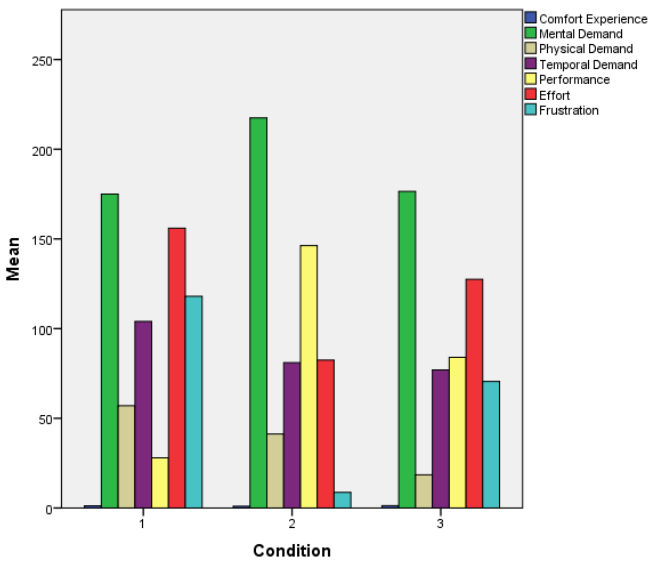


Figure 67: Descriptive Statistics for the NASA TLX between conditions

Table 23: Descriptive Statistics for the NASA TLX between conditions

Condition		Comfort Experience	Mental Demand	Physical Demand	Temporal Demand	Performance	Effort	Frustration
1	Mean	1.20	175.00	57.00	104.00	28.00	156.00	118.00
	N	5	5	5	5	5	5	5
	SD	0.45	120.00	95.56	134.79	26.60	73.77	121.17
2	Mean	1.00	217.50	41.25	81.00	146.25	82.50	8.75
	N	4	4	4	4	4	4	4
	SD	0.00	127.64	79.20	107.29	198.05	45.74	8.54
3	Mean	1.27	176.50	18.50	77.00	84.00	127.50	70.50
	N	11	10	10	10	10	10	10
	SD	0.65	143.28	35.91	81.66	57.53	83.21	110.62

Examining the results of happiness, participants appeared to recognize that condition three (M=4.86, SD=1.94) is happier than condition one (M=2.67, SD=2.45). These results may be attributed to the participants' feelings of accomplishment from the task and the amount of effort needed to accomplish the task. Based on the descriptive data from the NASA TLX shown in Table 23 and Figure 67, participants that performed the first condition felt they achieved a better performance level (M=28.00, SD=26.60) compared to the participants performing the third condition (M=84.00, SD=57.53). However, to accomplish the higher performance level, participants performing condition one put forth more effort (M=156.00, SD=73.77), felt more frustration (M=118.00, SD=121.17) and felt a higher temporal demand (M=104.00, SD=134.79) compared to their condition three counterparts for frustration (M=70.50, SD=110.62), effort (M=127.50, SD=83.21) and temporal demand (M=77.00, SD=81.66). Even though participants performing condition one appeared to have higher levels of performance, the higher levels of frustration and effort may have had a negative effect on the levels of happiness for condition one. These results may be attributed to the psychology of stress and frustration. The authors of [83] show that stress-induced release of endorphins actually helps mitigate the stress created by the task. However, there is an optimal level of stimulation where too much or too little stress can be detrimental [84]. For instance, even though participants performing condition one thought they performed better, the effort and frustration may be too high in order to warrant a happier response when experiencing VITA. The key is that the individuals must be able to cope with the stress. For instance, large job satisfaction comes with having a job/career where there is an optimal level of mental challenge with which the worker can cope [84]. In this case, there might be too much frustration, effort and temporal demand needed to achieve greater performance for the first condition compared to the third condition hence, condition three is a happier experience.

The results show that participants appear to find condition three sadder (M=2.39, SD=1.85) than condition one (M=1.07, SD=1.57) and condition two (M=1.17, SD=1.52). Figure 68 below shows that the happiest condition appears to be the saddest condition; the least happy condition appears to be the least

sad condition; and the condition in the middle, in terms of happiness, appears to be in the middle condition for sadness. It is surprising that the participants rate the happiest condition as the saddest condition because it was originally hypothesized that the happier a condition is, the less sad the condition will be. This was originally thought because sad music is shown to elicit sad moods, which are emotional markers for an unpleasant state of being [4] that activate markers in the brain for avoidance [4]. However, the authors of [4] discuss results of previous studies that reveal a link between happiness and sadness. In the studies, participants were asked to rate happy and sad feelings on two separate unipolar scales similar to the unipolar scales used in this thesis. The studies revealed that participants are ambivalent towards sad music. The authors also state that sad sounding music may have a cathartic effect that helps users to connect with their feelings, which allows their feelings to dissipate [4]. The results also show that people made to listen to music preferred the soothing effect to sad music [4]. This may make sad music pleasant and happy to experience..

In addition, the NASA TLX (see Figure 67 and Table 23) results may provide more insight as to why there are differences between condition three and two. Condition three may be the saddest because the participants felt they were the less successful in their performance of comprehending the music (M=84.00, SD=57.53) compared to participants performing condition one (M=28.00, SD=26.60). In addition, condition three may be sadder (M=2.39, SD=1.85) than condition two (M=1.17, SD=1.52) because participants felt they were less successful in comprehending VITA. The differences may be attributed to the levels of frustration experienced and effort required for each condition. Condition three was found to be more frustrating (M=70.50, SD=104.24) and required more effort (M=127.50, SD=83.21) compared to the level of frustration (M=8.75, SD=8.54) and effort for condition two (M=82.50, SD=45.74). The higher levels of effort and frustration may make the experience sadder for participants performing condition three.

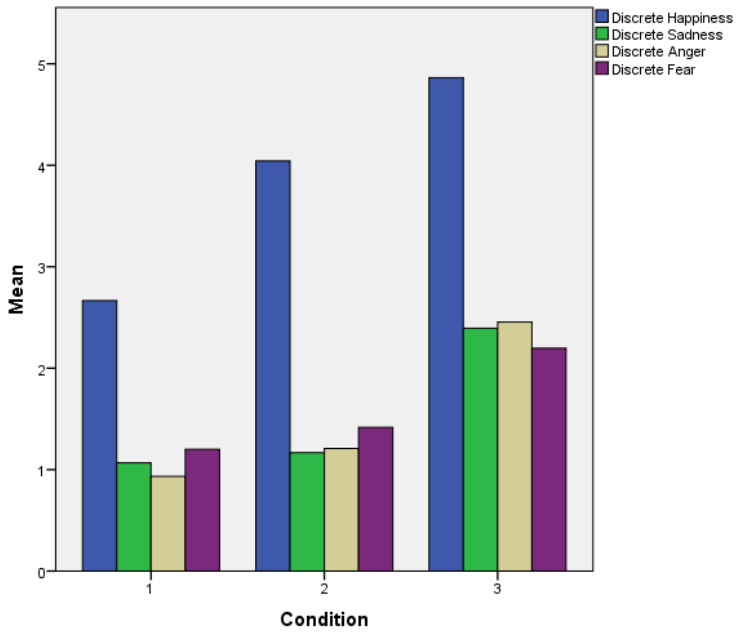


Figure 68: Discrete Levels of Emotions for each Condition

Examining the result of the levels of fear, participants appeared to recognize that condition three has a higher level of fear (M=2.20, SD=1.96) compared to condition one (M=1.20, SD=1.73). This may be attributed to the participants' comments for condition three. One participant stated that they did not like "so much going on without a warm-up [period] or previous knowledge of what to expect from either the visualization or vibration elements", this may have contributed to the fear factor of condition three. Another participant stated that they "were expecting a vibration to come that never did", another stated that the song "came as a shock at first" as well as another commented that "sudden stops [are] frightening (jumpy) when [they begin] again". These comments point towards combinations of elements that are disharmonious or a clash of what was happening and what was expected. This signifies that participants may have thought there was a lack of harmony when experiencing VITA, which creates a more fearful experience. [4] states that fear is created using dissonance and irregular rhythms. Even though fear is measured for this study, [4] states that there is a problem with fear measurement. The problem is that fear can be perceived and measured reliably when the stimuli are designed to specifically elicit fear; however,

the music chosen for this study was not selected for its ability or created to elicit fear. This may create accuracy problems when measuring fear. Also fear measurements may further be influenced by the levels of sadness experienced during the song. [82] shows that there was a link shown between sadness and fear, where sadness and fear influences each other.

The happiness, sadness, and fear results alongside the NASA TLX provide useful insight as to the workload that may be required to comprehend specific emotions in VITA. Composers and artists should take these results into account when creating music or playing music through VITA because certain playback conditions may influence the emotions experienced by the end users. The third condition-while conveying similar trends in discrete emotions as the other two conditions- conveyed happiness, sadness and fear to greater than the other conditions. In addition, composers and musicians must be cautious when attempting to elicit fear as the literature shows that it is difficult to measure fear in music [4]. However, it is recommended to use condition three as the playback scenario for VITA as it may maximize the experienced emotions.

4.2.1.2.3 Within Genres and Between Deaf and Hearing Groups

Examining the details of the discrete emotions for D and H groups, it appears that H participants perceive rock music as more angry (M=2.80, SD=2.35) than D participants (M=1.50, SD=2.32). These results can be explained by the trend shown in Figure 69. The trend may be attributed to intricacy of the instruments and bass components of each genre. D participants appeared to perceive jazz music as angrier than H participants. Jazz music contains intricate and complex melodies with a little to no bass line that is comparable to RHRB music. The intricate melodies without a bass component on the Emoti-Chair may make the vibrations harsher and therefore angrier. This is shown by the levels of agreement with the statement “the vibrations were too strong”, where jazz music (M=2.65, SD=0.49) portrayed stronger vibrations compared to pop music (M=2.76, SD=0.44). Pop music is not as intricate as jazz music and contains a stronger bass component, which may reduce the anger level in the song for both H and D individuals by not saturating the visualization and Emoti-Chair with as complex movement and

vibrations. However, the trend shows that as a bass component is added to the music and the intricacies of the movement of the music are reduced, D participants find it less angry whereas H participants find it angrier. This holds true for RHRB and rock music where D participants found RHRB and rock less angry than H participants found rock music. Hence the levels of anger may be related with the levels of bass and complexity of movements where a higher complexity of movement and less bass may make music less angry to H participants and angrier to D participants whereas lower complexity of music and more bass may make music angrier to H participants and less angry to D participants. This may be attributed to the participants' experiences with music. H individuals tend to rate RHRB and rock music as angrier genres compared to classical, country and jazz (see Table 24), whereas RHRB appears to be the most liked genre because of the bass. D participants may associate bass with a more pleasant experience. These results are useful to individuals that are interested in composing music through VITA. Composers should take into account-that because VITA conveys music through two modalities and each modality is different than conventional hearing music- that the combination of musical constructs may convey emotions differently than hearing music.

Table 24: Descriptive Statistics for the levels of anger for each hearing status and genre

Level of Anger for each Genre	Hearing Status	Mean	SD	N
Classical	Deaf	1.50	1.35	10
	Hearing	1.50	1.27	10
	Total	1.50	1.28	20
Country	Deaf	1.90	1.97	10
	Hearing	1.30	1.49	10
	Total	1.60	1.73	20
Jazz	Deaf	2.80	2.66	10

	Hearing	1.60	1.27	10
	Total	2.20	2.11	20
Pop	Deaf	2.10	2.13	10
	Hearing	1.20	1.23	10
	Total	1.65	1.76	20
RHRB	Deaf	1.60	2.12	10
	Hearing	2.10	1.85	10
	Total	1.85	1.95	20
Rock	Deaf	1.50	2.32	10
	Hearing	2.80	2.35	10
	Total	2.15	2.37	20

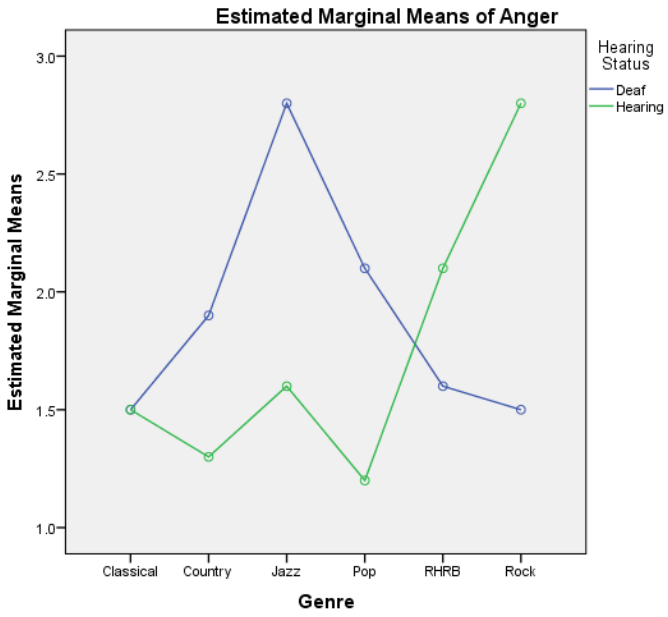


Figure 69: Descriptive Statistics for the levels of Anger between each Hearing Status and Genre

4.2.1.2.4 Comparison between Discrete and Continuous Emotions

The purpose of measuring emotions using the continuous and discrete emotional scales is to check if there is consistency in participants' ratings of emotion between the two emotional scales. It was hypothesized that the results would show statistically significance for the levels of arousal and valence in order to validate the results alongside the discrete emotions but this was not the case. There were no statistically significant results pertaining to the level of valence and arousal in order to make definitive conclusions to the experience of happiness, sadness and anger using the continuous model. However, the results of the discrete emotions did provided statistically significant results into the emotions experienced by the participants.

These results show that the participants may have been confused when rating their emotional experiences on the 3D continuous scale as it is more confusing to rate happiness using valence and arousal when the study is not designed to specifically test for arousal and valence as in [2]. Also, [2] show that the continuous scale is more difficult to differentiating between emotional such as sadness and fear that have similar valence and arousal but differ in the amount of tension. The discrete emotional scale is used in hearing studies to show participants' ability to report experienced emotions in voice and in music [2] and may not be suited to rating visual and vibro-tactile music as the mediums are different and unconventional compared to auditory based music. Also, there is little research pertaining to the use of the 2D continuous scale in experiencing emotions in vibro-tactile and visual music. Hence the results show the discrete unipolar scales may be easier to relate because they are listed as the emotions and are spoke of in colloquial language versus a psychological model, which may have been confusing to individuals. This is a problem that has surfaced throughout the study; future studies should take into account of the language used as psychological terms. Also, future studies into visual and vibrotactile music should use the discrete unipolar emotional model to measure emotions as these results suggest it is a better model for participants to relate to when experiencing music in an unconventional way.

4.2.1.3 Levels of Distraction and Focus

4.2.1.3.1 Within Genres Chi-square

Examining the details of the levels of focus on the visualization, participants appeared to be the most focused on pop (M=2.47, SD=0.84) and jazz (M=2.35, SD=0.93), and the least focused on RHRB (M=2.20, SD=0.95) and rock (M=2.15, SD=0.10). In addition the non-significant results of classical and country music may indicate a level of confusion or boredom when viewing these genres, which may be attributed to their characteristics. Country music is characterized by guitars with little or no drums. This may not create the most visual movement and therefore might create boring visualization to view thus leading to a more distracting visualization. Opposed to this, the classical genre has more expressive music played by large orchestras and more instruments than other genres in this study. This may create too much visual movement that may be confusing to the participants, which may create a boring experience.

Even though the results show that participants focused on the visualization (2.00<=M<=2.47), the results relating to the levels of focus on the vibro-tactile component show that participants focused more on the vibrations (2.40<=M<=2.65). This is not surprising as the vibro-tactile feedback is more difficult to ignore compared to visuals, which individuals can look away from. These results point towards participants relying more heavily on the vibrations for an emotional experience of the music compared to the visualization. Participants may use the visualization component as a helper component for learning the intricacies of the vibrations in more complicated songs. In addition, it was surprising that participants were most distracted when experiencing RHRB (M=2.40, SD=0.88) and jazz (M=2.40, SD=0.88) music. This is surprising because it was hypothesized that the vibro-tactile component of RHRB music would be the strongest such that participants would be the most focused. This may be because, even though RHRB contains a strong bass line, a large portion of the bass line comes from the vocals, which are missing in MIDI music [10].

4.2.1.4 Levels of Agreement

4.2.1.4.1 Between Genres

Examining the details of the levels of vibrations for each genre, participants agreed that the levels of vibrations for country music was stronger (M=2.24, SD=0.90) than those for jazz (M=2.65, SD=0.49) and pop music (M=2.76, SD=0.44). A possible reason is that country music contains fewer instruments and has little to no bass/beat compared to pop music, where pop music contains a more prominent bass line. The prominent bass line present in pop provides deeper vibrations along a larger area of the user's back, which may closer imitate the bass feeling provided by a subwoofer and may be less irritating to feel compared to the bass-less feeling given by country music. When there is no bass, the resulting vibrations may provide constant irritating stimuli to the skin. This may be similar to how H individuals perceive high-pitched sounds. Loud and high-pitched music can be irritating to listen to as it is grating to the listener's ears. For example, one participant stated that they "didn't like the long vibrations that were in the same area. It felt annoying and uncomfortable after a while" and other participants stated that they were expecting the vibrations to encompass their entire body similar to how a subwoofer works.

However, the bass line provided by pop music may be deeper and similar to vibrations from a subwoofer. For instance, a participant stated that the deep vibrations reminded them of the body shaking vibrations one feels when accelerating quickly in a fast car. In addition, bass provides the on/off stimulation as the beat. The lack of a bass line in country music may make it unfamiliar to participants, where some participants were "expecting vibrations to come that never did". A participant also stated that the unexpected "sudden stops [are] frightening", which may make harsher vibro-tactile experiences. Even though participants rated the vibrations as strong, they did not rate them as uncomfortable. On the post study questionnaire participants rated their overall experience as comfortable (M=1.2, SD=0.52), where a rating of 1 was comfortable).

In addition, while examining the results, participants appeared to find the levels of vibrations for classical music (M=2.76, SD=0.56) just as weak as they were for pop music (M=2.76, SD=0.44). This is surprising because classical music is performed by orchestras with a plethora of instruments that can provide an overwhelming emotional and auditory experience. In addition, it was thought that the large amounts of instruments in classical music would have made the vibrations stronger because it would create more vibrations; however, this was not the case. This may be attributed to the style of music. Classical music, although containing many instruments that play in unison, tends to have more complex dynamics, where a single musical composition may become louder (crescendo) and become softer (diminuendo). In addition, there are many different words to denote different dynamics in classical music such as calando, calando, dal niente, fortepiano, fortissimo piano, etc. The greater complexity of classical music may have created vibrations that vary from soft to hard unlike pop music, which tends to be consistently loud and strong.

Even though the purpose of VITA is to convey music through visual and tactile domains, and it is shown those participants find the vibrations in certain genres stronger than others, it does not indicate a negative impact when having strong vibrations. However, the results indicate that, when composing music for VITA, it would be beneficial to include a stronger bass line mixed with other instruments to minimize any annoyingness from high-pitched and loud signals thus maximizing the enjoyability. This may provide a more familiar musical experience mimicking subwoofers. Also participants stated that they wished to have the bass line turned louder as to have their body shake more. This should be taken into account when composing or playing music through VITA in the future.

4.2.1.4.2 Between Conditions

As a reminder, participants were asked to rate their level of agreement with each of the following seven statements on a 3-point Likert-Scale ranging from “1-Agree” to “3-Disagree”. The seven statements are as follows. The following section will refer to each statement by its number. The levels of agreement for

statements 1 to 4 were significant between conditions one and three (see Table 67 and Table 68). 'Understand' was described to mean the ability to experience emotions from the system.

1. The movement was hard to watch
2. The patterns and shapes on the screen were pleasant to watch
3. The vibrations helped me understand the visualization
4. The visualization helped me understand the vibrations
5. The vibrations were too strong
6. The vibrations up and down my back felt pleasant
7. The pattern of vibrations on my back was confusing

Examining the results of the participants' levels of agreement for statement 1 to 4, participants appeared to recognize that the movement in condition three (M=2.61, SD=0.74) is easier to view compared to condition one (M=2.03, SD=0.10); participants appeared to recognize that the patterns and shapes are more pleasant to watch for condition three (M=1.47, SD=0.81) compared to condition one (M=1.97, SD=0.93); and the vibrations helped more when attempting to experience the visualization and the visualization helped more when attempting to experience the vibrations for condition three (M=1.47, SD=0.81) (M=1.48, SD=0.83) compared to condition one (M=2.14, SD=0.97) (M=2.21, SD=0.96). Overall, the participants appeared to recognize that experiencing VITA with the third playback condition is more useful. It was original hypothesized that the movement would be harder to watch for condition three, and therefore the patterns and shapes less pleasant to view. This was originally thought because the visualization for condition three conveys more information than condition one; however, this is not the case. In addition, it was originally hypothesized that the vibrations in condition three would provide users with a worse experience because it would overwhelm the participants' senses; however, this was not the case. These results may be attributed to the NASA TLX descriptive shown in Figure 67 and Table 23 as well as the participants' comments. In terms of the NASA TLX results, the levels of frustration (M=118.00, SD=121.12) and effort (M=156.00, SD=73.77) are higher for condition one compared to

condition three's frustration (M=70.50, SD=110.62) and effort (M=127.50, SD=83.21). The lower levels of frustration and effort may make condition three easier and more pleasant to watch compared to condition one. Also the reasons for the vibration helping the visualizations and the visualizations helping the vibrations for condition three may be attributed to the characteristics of each condition. As previously mentioned, condition three plays the entire signal on both devices, whereas condition one plays percussion on the chair and instruments on the visualization. The participants commented that they attempted to see what altered in the visualization to learn the Emoti-Chair and vice versa; however, this was more difficult when percussion would not appear on the screen and instruments would not appear on the chair. Participants performing condition one commented that there "[was] a lack of synchronization with the vibrations", they did not like "when the chair did not move" and this happened often during genres that contained less percussion instruments. These results indicate that condition three is a better condition for learning from each device as sensory redundancy increases robustness of learning. However, based on the chi-square results from the post-study questionnaire, it is promising that no matter what condition the participants performed, participants agreed that the vibrations helped them experience the visualization (M=1.20, SD= 0.62) and that the visualization helped them experience the vibrations (M=1.60, SD=0.88). The goal of VITA is to provide an informative and entertaining music experience system where individuals can access the emotional aspects of music. These results indicate that condition three provides this experience to its users with less effort and less frustration than the other playback conditions. This is because condition three creates a visual experience that is easier and more pleasant to watch to consume music, and provides a better environment for the user to learn the vibro-tactile component using the visual component and vice versa.

4.2.2 Post-study Discussion

4.2.2.1 Levels of Enjoyability

4.2.2.1.1 Chi-Square

Examining the results, it appears that a majority of participants, regardless of genre, condition and hearing status, enjoyed (M= 2.85, SD = 0.49) experiencing music with VITA (see Figure 70). These results indicate that music composers can be confident that they can provide an enjoyable experience, when composing music for the VITA system. Also, these results show that end-users can be confident they will have an enjoyable experience when using VITA to experience music.

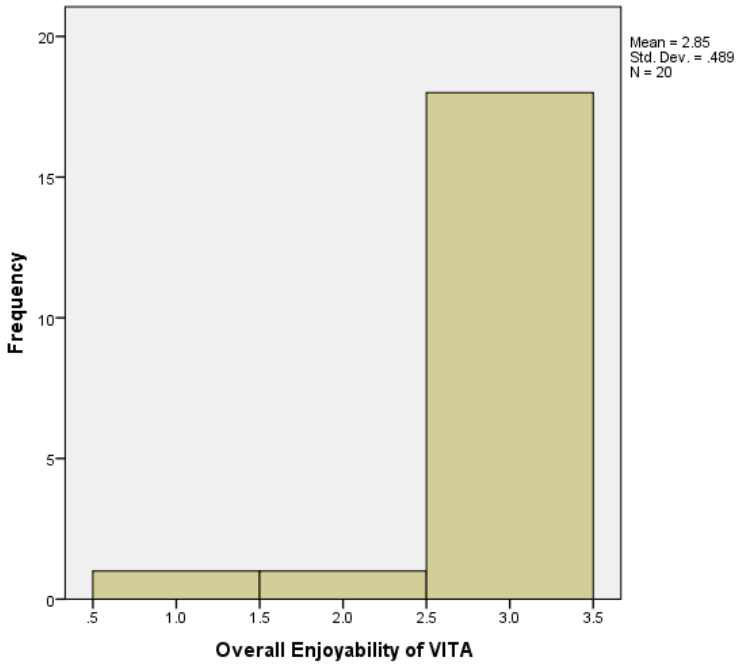


Figure 70: Overall Enjoyability of VITA regardless of genre, hearing status and condition

4.2.2.2 Levels of Agreement

4.2.2.2.1 Chi-Square

As a reminder, participants were asked to rate their level of agreement with each of the following seven statements on a 3-point Likert-Scale ranging from "1-Agree" to "3-Disagree". 'Understand' was described to mean the ability to experience emotions from the system. The questions are:

1. The vibrations helped me understand the visualization
2. The visualization helped me understand the vibrations
3. I was distracted throughout the experience
4. The experience was comfortable

Examining the results of the levels of agreement with each of the previously mentioned statements, participants appeared to agree that the vibrations helped them experience the visualization (M=1.20, SD=0.62) as well as they appeared to agree that the visualization helped them experience the vibrations (M=1.60, SD=0.88). These are promising results as it was originally hypothesized that participants would using the visualization component to help them learn the vibro-tacticle component and the opposite was also thought. These results show the same phenomena described in [6], where sensory redundancy is used by people to increase their robustness of learning. These results show that the end-users can use VITA to gain a greater emotional experience of the music. Also, it is promising that the participants appeared to be focused throughout the experience (M=2.60, SD=0.82) as well as find the experience comfortable (M=1.20, SD=0.52). This indicates that the system is viable for long-term use by individuals to experience music in their home for relaxation and entertainment. In fact, participants showed interest in wanting to buy a commercialized system for home use.

4.2.2.3 Answers to Questions

One set of questions in the post-study questionnaire pertained to what the participants thought the visual constructs in MusicViz meant. The three questions are shown below. As a reminder, the participants were

asked to select their answers from a list with 5 choices. The choices are: (1) intensity; (2) frequency; (3) beat; (4) grouping; and (5) other.

1. What does the brightness in the visualization mean to you?
2. What do the shapes in the visualization mean to you?
3. What do the different heights of the pipes on the screen mean to you?

4.2.2.3.1 Between Deaf and Hearing Groups

The results for the first question for each hearing status are shown in Figure 55. The results show that a majority of H participants (7/10) appeared to recognize that the brightness in the visualization corresponds to the intensity of the music compared to H participants that thought the brightness signified frequency (1/10), beat (1/10), instrument grouping (1/10) or other (1/10). The results from the D participants appear to be unsure as to the meaning of the brightness. An equal amount of D participants appear to think the brightness signifies intensity (3/9) and brightness (3/9). This is compared to the D participants that thought brightness signified beat (1/9), instrument grouping (1/9) and other (1/9). These results could be explained by the model of music that each hearing status has developed over their lifetime. H participants that have attended a public or private formal school system have taken at least basic music classes and have been introduced to music terminology. Also, all H participants answered to listening to music every day. These participants have most likely been introduced to colloquial terms in English such as a bright shirt being called “loud” or a bright colour being referred to as “intense”. Not all D individuals may have had the same music education in a public or private education system. In addition, depending on the level of their deafness, it is possible that individuals may have not been exposed to these English terms as many participants used ASL as their primary language, where English and H terms may not be used. Although, it is promising that many participants can recognize that brightness signifies the intensity of the music (10/20). Another possible confusion is that these terms may be vague, too complicated, too hearing-oriented or abstract for D individuals, even though the terms were created to be as least hearing-oriented as possible. It is recommended that further studies asking

participants to identify the meaning of specific visual constructs use different language that is more D oriented.

The results for the second question for each hearing status are shown in Figure 56. The results show that a larger consensus of H (5/10) participants appeared to recognize that the shapes signified the beat of the music. This is opposed to H participants that thought the shapes signified intensity (0/10), frequency (2/10), instrument grouping (1/10) and other (2/10). The results pertaining to the D participants also show that a consensus thought the shapes signified the beat of the song. This is compared to the amount of D participants that though the shapes signified intensity (0/9), frequency (2/9), instrument grouping (1/9) and other (1/9). Unlike the previous results, the results of D and H participants are similar and seem to show a similar trend. It was originally hypothesized that more participants would choose instrument grouping over everything else. This is because the visual component was designed to have similar instruments represented by similar shapes. For example, percussion drums are represented by toroid objects and all other instruments are portrayed as pipes. The results do not follow this hypothesis. These results are surprising because a total of two participants appeared to recognize that the shapes signify the instrument groupings whereas when the participants were asked to provide written answers for "what did the circles mean to you" and "what did the pipes mean to you", a total of 11 participants listed the circles as the beat and the pipes as "guitars", "piano", instruments, other tones, guitars, etc. These results point towards using shapes for groupings; however, participants did not choose instruments groupings from the multiple choice. These results may be attributed to the type of music the participants frequently access. Based on the demographics, 8/10 H participants enjoy RHRB music and 7/10 H participants enjoy rock music. This is opposed to the amount of H participants that listen to classical (1/10), country (1/10), metal (2/10) and other (2/10). The trend in the demographics shows that H participants tend to listen to music with a stronger bass and percussion orientation. This may have been driven to look for a bass component, which is a familiar and identifiable characteristic from H music. Even though D music is heavily reliant on bass and rhythmic gestures to RHRB-style music, it is intersecting that only three D participants

reported listening to RHRB music whereas five D reported listening to rock music, although this is a total of eight D participants that listen to music with a strong bass and percussion line. This may have also driven D participants to search for visuals that signify a bass-line. Overall, these results point towards more confusion as to what the visual constructs in VITA meant. This was a problem due to the obscure and abstract language used to refer to musical constructs. Initially it was thought that the abstract language would be less hearing biased; however, the language seemed only to confuse participants. Therefore it is important in future studies to use clearer language.

The results for the third question for each hearing status are shown in Figure 57. The results show that a larger consensus of H participants recognize that the heights of the pipes signify the frequency (5/10) of the music. This is compared to H participants that though the heights signified intensity (3/10), beat (1/10) and other (1/10). There was not a consensus as to the meaning of the shapes in the visualization for D participants. There was a tie between the amount of D participants that though the shapes signified the intensity (3/10) and frequency (3/10) as well as a tie between beat (2/10) and other (2/10). Even though a larger consensus of H participants chose frequency as their answer compared to the other answers, the results show there may have been some confusion as whether the answer should be intensity or frequency. This is also true with D participants' answers, where D participants may have had a more difficult time choosing between intensity and frequency. This may be attributed to the same reasoning as presented in the explanation for the results of the first question. H participants may have been exposed to musical terms such as pitch height, where terms such as 'higher frequency' and 'higher pitch' may be more common place terms in hearing English culture, whereas these terms may not be used often in ASL. Future studies should be more careful as to choose language this is less obscure and abstract in order for participant to more accurately identify the visual constructs without inhibiting their ability to answer the question with obscure language.

4.2.2.3.2 Between Conditions

The results for the first question for each condition are shown in Figure 58. The results show that larger consensus of participants that performed condition three thought the brightness signified the intensity (6/11) of the music. This is opposed to participants that performed condition three that thought the brightness signified the frequency (2/11), beat (1/11), grouping (1/11) and other (1/11). Participants that performed condition one thought the brightness signified intensity (2/4) compared to beat (1/4) and instrument grouping (1/4) and participants that performed condition two thought that the brightness may signify intensity (2/4) and frequency (2/4) equally. The results seem more confusing for participants that performed condition one and two because there is no consensus as to what brightness may represent in the visualization. This may be attributed to the characteristics of each condition. Condition one and two split the instruments from the percussion and designate either the Emoti-Chair or MusicViz for playback depending on the chosen condition, whereas condition three plays the entire signal concurrently on the Emoti-Chair and MusicViz. Participants also commented that that they used the vibrations from the Emoti-Chair and the visuals from MusicViz to learn how each device is conveying the signals; however, it may be more difficult to do when the Emoti-Chair and visualization are not conveying the entire signal. This may have confused the participants' decisions that performed condition one and two, whereas the participants performing condition three reached consensus that the intensity is represented by the brightness in the visualization. However, previous results from the D and H groups point towards obscure and confusing language as the culprit for the confusing answers. Future studies should make sure use language that clearer and less abstract such that there is less of a language barrier when comprehending the questions.

The results for the second question for each condition are shown in Figure 59. Participants performing condition three were in consensus that the shapes signify the beat of the music (6/11) as were the participants performing condition two (3/4). The results of condition one signify some confusion. Participants performing condition one thought the shapes signified frequency (2/4) and other (2/4). Once

again, these results may be attributed to the nature of each condition. Condition three provides a complete signal to the Emoti-Chair and Visualization that is missing in condition one and two. This may provide enough information to reach a surer conclusion as to what the shapes meant. These results are surprising because MusicViz was designed to have similar instrumented grouped together by similar shapes and even though they recognize this in their written descriptions, they do not recognizing this when answering this question. This could be due to the abstract language used in the questions. In addition it could be attributed to interactions between the Emoti-Chair and visualization. D and H participants are more inclined to listen to music that is more bass oriented, which is strongly felt by each hearing status and seen as an integral part of the genres they listen to. In addition, the lower frequencies from the Emoti-Chair could overpower the higher frequencies, thereby making it simpler to perceive the percussion and create a mental link between lower vibrations and the percussion instruments on the screen.

The results for the third question for each condition are shown in Figure 60. The results show that a larger consensus of participants that performed condition three appeared to recognize that the heights of the pipes signify the different frequencies of the music (6/11) compared to intensity (2/11), beat (1/11) and other (2/11). The results also that a larger consensus of participants that performed condition two thought the heights of the pipes signified the intensity of the music (3/4) compared to other. Lastly, the participants that performed condition one thought the heights of the pipes signified either frequency (2/5) or beat (2/5) and one participant though it signified the intensity. It was originally hypothesized that more participants would choose frequency. This was thought because MusicViz was designed using the psychological model described in the literature review. However, this is not the case and a closer look at the conditions provides an explanation. It would be impossible to identify the meaning of the heights of the pipes in condition two because MusicViz does not render the pipes when playing the second condition; MusicViz renders only toroid objects, which represents the percussion.

4.3 Limitations

As with any study, there are a number of limitations. Some limitations pertain to the methodology and some pertain to the technical limitations imposed by a timeline. In this study, the technical limitations pertain to the technical limitations of MusicViz, the limitations of the Emoti-Chair prototype utilized, the lack of the number of participants, the complexity of language for Deaf participants and difficultly with GSR equipment. These limitations are discussed below.

4.3.1 MusicViz's Technical Limitations

There are technical limitations with MusicViz that were unavoidable due to the timeframe required to complete the project. These limitations pertain to the dated 3D graphics, lack of complex animations, lack of stereoscopic 3D, lack of a changeable colour pallete based on psychological perception of colour and MIDI.

4.3.1.1 Dated 3D graphics and Lack of Complex Animations

A large chunk of the technical limitations were due my lack of knowledge of computer graphics. Initially, the music visualization was created using 2D flash animations; however, the visualization technique did not meet expectations in studies and it was hypothesized that a 3D visualization may improve the enjoyability and the visualization's ability to convey information about music. When I chose to tackle this problem of communicating music to the hearing impaired, the initial problem I encountered was learning 3D computer graphics. I had to quickly learn the mathematics and the basic techniques involved considering it is impossible to jump into basic rendering and graphics engines without the foundation. During the process of learning computer graphics in OpenGL, I developed a basic animation system that eventually became the basis of the animations in MusicViz. In addition, the shading technique in MusicViz utilizes the basic Goraud shading technique that is available in OpenGL.

The basic shaders built into OpenGL do not allow for more complex lighting and texturing techniques that are staples of modern real-time graphics. Also, OpenGL does not provide animation systems for

particles, characters, or body movement. This is because OpenGL is not a graphics engine. OpenGL is a library on which graphics engines -that contain various complex animation, particle modeling, and advanced global lighting systems- are built. This makes MusicViz's graphics dated when compared to modern real-time animation and rendering engines.

4.3.1.2 Lack of Stereoscopic 3D

The participants commented throughout the study that the quality of 'immersiveness' can be improved with the use of stereoscopic 3D glasses. Some participants even mentioned that they initially thought the music visualization would be in stereoscopic 3D when making the decision to participate. Therefore the quality of the visualization may have been improved if stereoscopic 3D was implemented.

4.3.1.3 Lack Changeable Colour Palettes

One of the initial plans with the visualization was to incorporate different colour palettes into the visualization that would change to alter a specific mood that is dependent on the genre being played. It was quickly discovered when writing the literature review that this is an extremely difficult task due to the wide range of meanings of colours across cultures and even within similar cultures (see literature review); and considering the goal of MusicViz is to utilize pre-attentive visuals to convey music, which requires the least amount of thought to extract the most meaning, colour would be a poor choice. In addition, participants commented that even though the colours were enjoyable, it was preferred to change the colours of each visual construct to colours based on personal preferences.

4.3.2 Emoti-Chair's Limitations

There were unavoidable difficulties with the Emoti-Chair's software. The software used to drive the Emoti-Chair was developed in Max/MSP as a prototype and proof-of-concept for a commercialized product, which had a slight lag effect. This delay is difficult to notice but large amounts of stimulus on the screen and Emoti-Chair may have made the lag more noticeable. One participant stated that "at times it

felt that the visuals and tactile [were not] synchronized"; however, it is interesting that only participant commented on this issue.

According to the research performed by Branje, the FDLs of human skin is 200 cents, with a maximum of 1000Hz; however, auditory music contains frequencies above 1000Hz. Emoti-Chair users are missing 19,000Hz of the music. The VITA system suffers from this problem.

4.3.3 Limited Number of Participants

The quality of the study could have been improved with an increased amount of participants and a wider range of age groups. A larger group of individuals might provide an insight as to how different age groups use VITA to experience music and provide a greater insight to the effects of the different conditions, each hearing status and each genre on the enjoyability, emotions and focus on the VITA system. In addition, there was not enough variability in ages to do age statistics. The sample size favoured younger people, which have a tendency to show preference towards RHRB and pop and newer types of rock. This may have skewed results.

4.3.4 Complexity of the Language in the Questionnaires

The questionnaires were designed to be as easy as possible for D and H participants to comprehend and answer; however, the language used in the questionnaires was still too complicated for some D participants. One problem was that it was difficult you diverge from hearing terms when referring to music. A solution to this was to user more abstract terms and phrases; however, this overcomplicated the language for H people and did not achieve the effect of simplifying the language for D participants. An inherent problem with questionnaires for D participants is that their education tends to be less comprehensive than their H counter-parts. This introduced some complexity even when interpreted into ASL.

In addition, participants were asked to rate their level of agreement with specific statements. The statements could have been phrased in neutral, positive or negative terms instead of alternating between each for different questions. .

5 Conclusions and Future Work

5.1 Conclusions

A system called VITA was designed to provide cross-modal access to music through the tactile and visual domains. VITA is composed of two individual systems that are combined. The first system, MusicViz, is a real-time and automated music visualization system designed to translate auditory cues to visual cues using a psychological model of auditory and visual perception of sound. This is a novel approach to creating music visualization systems as many do not have a psychological foundation to justify their visuals. The Emoti-Chair is a haptic feedback chair that is used to translate auditory cues to the tactile domain. A user study was conducted on this system. From this research, it was found that it was feasible to provide music to individuals through the use of a VITA.

Regarding the effects of the genre of music on the enjoyability and emotions experienced, participants found all genres enjoyable to regardless of the genre experienced; however, there was may have been some confusion as to the enjoyability of classical music as it is an inherently more complicated genre due to the large instrument set used. In addition, the emotional responses rated on a multi-dimensional scale of emotion and the discrete unipolar scales follow similar results found in psychological research. Participants found that as a song becomes more exciting, it becomes more overbearing. Also, it appeared that pop music was the most enjoyable genre because of the higher levels of valence and arousal; psychological research shows that levels of enjoyability are linked to higher levels of valence and arousal. It also appears that a lack of a percussion line in music may have an effect on participants' perceptions of the strength of the vibrations in general. Participants found that country music had the strongest vibrations; however, it has some of the least complicated movement in terms of the amount of

instruments. This may be attributed to the fact that D individuals tend to focus on the bass-line as the primary source of music and there is none in country. This may make it irritating to participants.

Even though both H and D participants found the experience enjoyable regardless of the condition and genre experienced, H participants appeared to enjoy VITA more. This was not expected because accessing music through visuals and tactile without sound is an alien idea for most H individuals, whereas D individuals are more attuned to this. The higher enjoyability level may have been attributed to a novelty effect for the H participants compared to D participants that already had an idea of what they would have liked to see in music visualizations.

The research found that there was a statistically significant interaction effect between hearing status and genres. H participants found rock music angrier than D participants. This hints at a greater trend where H participants find genres with a smaller bass line as less angry. As a bass line is introduced, H participants found it angrier. The opposite was seen for D participants. This may be because D participants expect to be vibrated by their music. Many participants wished they could get the same experience that they receive when at home or driving where their whole body vibrates to extremely loud music; however, this is not possible with the Emoti-Chair. This may influence their anger level where music with a smaller bass line irritates them because the experience did not follow their initial expectations as to what music feels like.

The research found an interaction effect for the levels of distraction from the visualization between each hearing status and genres. The results show that D participants were more distracted from the visualization when viewing RHRB music than their H counter-parts. It was a surprising result because D individuals prefer RHRB music because of the strong bass line. A possible reason was that D individuals response visual irregularities, which RHRB does not provide because of its repetitive nature. However, in terms of the level of focus on rock music, the opposite was shown, where D participants were more focused on the visualization than their H counter-parts. This may have occurred because rock music provides the much needed irregularities to create more diverse movement that catches D each

participant's attention, whereas H participants may have paid less attention, whereas the visual irregularities were complicated for H individuals.

The descriptive statistics found that that a majority of H participants were able to correctly identify that brightness signified intensity, whereas D participants were confused. This was attributed to the abstract language used to identify brightness as intensity. Even though brightness is synonymous with intensity in spoken languages, it may not be as widely used in Deaf culture. However, neither H nor D participants were able to correctly identify the shapes in the visualization as instrument groupings. This was attributed to the language used in the multiple choice questionnaire because when asked for a written description, participants were able to correctly identify that the different shapes signified instrument groupings. The use of the word 'shapes' may have been too general and too abstract for participants to identify the correct answer on the questionnaire. Lastly, when participants were asked the meaning to the heights of the pipes in the visualization, a larger amount of H participants understood that the heights signified the frequencies in the music. D participants' answers point to confusion. Again, this may be attributed the language used in the questionnaire because terms such as 'higher frequency' and 'pitch height' may be more common place in spoken language compared to ASL.

Although there are differences in the levels of enjoyability, focus and anger between hearing status and genres, the study results show that no matter the genre experienced by either hearing status, VITA is an enjoyable experience.

The study found there were statistically significant results due to the conditions randomly assigned to each participant. Condition three was shown to be more overbearing than condition two because condition three saturates the Emoti-Chair and MusicViz with more information. The larger amount of information makes condition three more overbearing as opposed to condition two, where the audio signal is spatially separated between the Emoti-Chair and visualization.

The conditions assigned also had an effect on the happiness, sadness and fear ratings. Condition three was found to provide not only a happier but also sadder, and more fearful than condition one. [4] show that this is a valid response to music, where happiness and sadness is linked.

The results show that there were differences between conditions one and three as well as differences between condition one and two for the levels of focus on the Emoti-Chair and visualization. Participants focused more on condition three compared to condition one. In addition, participants performing condition one were more focused on the Emoti-Chair and visualization compared to participants performing condition two. There were no statistically significant differences in the levels of focus on the Emoti-Chair and visualization for condition two versus three.

Overall condition three was seen by participants as the most useful condition for enjoying and comprehending music in VITA. Participants appeared to recognize that the movement in condition three is easier to watch compared to condition one. Participants appeared to recognize that the patterns and shapes are more pleasant to watch for condition three compared to condition one. Lastly, the vibrations helped more when attempting to experience the visualization and the visualization helped more when attempting to experience the vibrations for condition three compared to condition one.

The conditions assigned to each participant also had an effect on the participants' answers as to what the visual constructs in the MusicViz signified. Participants that performed conditions one and two were unable to correctly identify the significance of the brightness, the shapes and the heights of the pipes in the visualization.

Although there are differences in the levels of overbearingness, happiness, sadness, fear, and levels of focus on the visualization and Emoti-Chair and levels of agreement between conditions, the results show that the system provides an enjoyable experience regardless of the assigned playback condition.

VITA is not without its limitations and there is a plethora of future work to improve the usability of the system. However, it is promising that this system is able to convey some emotions and provide an

enjoyable experience unlike previous music visualization systems in commercial and educational domains. D participants even commented that they would be interested in purchasing the VITA system for personal use. The founder of TAD Inc. at Ryerson's DMZ has expressed interest in offering MusicViz alongside the commercial version of the Emoti-Chair.

In summary, the results of the user study into the feasibility of presenting music through VITA suggest that the experience is not only enjoyable and entertaining but also informative. VITA allows for the automatic and real-time playback and conversion of auditory music into vibrations and visuals based on a psychological model of the perception of sound. The purpose of the study was to discover not only the differences between genres, hearing status and conditions but also to find the best possible playback condition for the music visually and tactually. Participants, regardless of their genre preferences, hearing status and condition viewed, found VITA an enjoyable, useful and entertaining method of experiencing music. Participants stated that they were able to pick out subtle changes in the music and the system actually conveyed meaning. This was corroborated by the results, which showed that participants' emotional responses –measured in the multi-dimensional and unipolar scales followed similar emotional results from hearing music based studies and points toward VITA providing an entertaining, useful and informative music experience system. Overall, condition three was shown to be the most useful playback condition for experiencing music through VITA. Conditions one and two spatially separate the instruments for playback on either the Emoti-Chair or MusicViz. This may have caused a disconnect from what was seen and what was felt, which made the experience confusing, less enjoyable and less informative because participants could not correlate the visualization with the vibrations and vice versa.

5.2 Future Work

5.2.1 Research Suggestions

1. A problem is that this study contained no objective measure of emotion. A possible avenue for future work would be to incorporate ECG (electrocardiogram) or EEG (electroencephalography)

to measure heart rate or the current flowing through the brain's neurons for the use of collecting users' reactions to stimuli.

5.2.2 Technical Suggestions

1. The VITA system is designed using two different development environments. MusicViz is programmed using Java and JOGL for easy interoperability between Mac, PC and Linux environments running a 32-bit or 64-bit Java installation and the Emoti-Chair's software is designed in Max/MSP. Even though MusicViz can run on multiple operating systems, it is restricted to desktop or laptop machines. Participants commented that even though they cannot use the Emoti-Chair while outside their homes, it would be a very good feature to load songs into their iPhones, iPads and Android tablets to visualize their favourite songs when commuting. Another possible avenue for future work is to offer MusicViz as a complete bundle with the commercialized Emoti-Chair system offered by TAD Inc.
2. MusicViz is developed as a proof-of-concept into a music visualization algorithm based on psychological research for my master's thesis. MusicViz does not attempt to create a graphically advanced visualization system by including high-tech features, which commercial graphics engines contain. However, considering there is positive results from the user study carried out, future-work is the potential commercialization of the product. The Emoti-Chair is already commercialized by TAD in the DMZ and MusicViz is paired with the Emoti-Chair for this thesis. It is a logical step to provide MusicViz as a commercialized bundle with the production-level Emoti-Chair. However, commercialization consists of translating a proof-of-concept thesis into a viable product. This is done by providing up-to-data graphics capabilities, which a commercial graphics engine provides. This allows for more complex animations and movement and design customizability by a graphics designer. In addition, modern smart-phones, tablets, desktop computers and even laptops have the ability to run modern graphics engines that can power

greater visuals. In addition, modern graphics engines can be utilized to offer the immersive 3D experience through stereoscopic 3D techniques.

3. Another issue with the current version of MusicViz is that it relies on MIDI music to provide the music data. As previously mentioned, MIDI is essentially sheet music for a computer. MusicViz utilized MIDI information to perform the translation to visuals. The problem is that MIDIs used in this study were created by amateurs. In addition, a majority of the songs existed as guitar tabs that had to be converted to MIDI using a specialized program. For MusicViz to be an end-user experience, this step must be removed such that an individual can load their song into the program. Two methods can be used to accomplish this task. One method is to perform wave-form analysis to extract pitch, tempo and amplitude for each individual instrument. This is another research topic that is still largely under developed. The other method is to utilize similar method as the creators of the game "Guitar Hero". The creators used MIDI signals incorporated into the wave-form music to signify when the player should activate the instrument. More MIDI notes were added as the difficultly was increased. This would be the most viable option such that the MIDI can be incorporated with the MP3 file for playback.
4. The BRASS system by Hiraga states that they include a *focus+context* view of the system. The VITA system provides a method to separate percussion from other instruments; however, providing a method to focus on specific instruments may provide greater control and increase the reported emotional experience that participants receive from the music and its components. Participants have commented that it would be a good feature to pause the song and move back and forth between different times in the song. It was also suggested to have the ability to haze-out specific instruments in the visualization and Emoti-Chair to focus on specific instruments.
5. [46] present a visualization method, which utilizes musical dyads and keys to colour the visuals. The visualization's colour differs depending on the tones played as well as the key in which the song is composed. Different keys are assigned to different colours on a circle-of-fifths wheel. At the moment, MusicViz has a predefined set of colours, which do not change. It may provide more

entertainment to colour the instruments based on the key of the musical piece. This may provide a psychological justification for a colour scheme.

6. Currently MusicViz synthesizes the music from the MIDI messages and the EMoti-Chair processes the resulting auditory signal to vibrations. However, the Emoti-Chair has a limitation where all sounds over 19,000 HZ are lost due to the mechanical limitations of human skin. A possible solution for future work is to perform a range conversion from MIDI pitch bytes (0 to 127) to frequencies values between 20Hz and 1000Hz. However, care must be taken because the issue with performing a range conversion is that the skin may not be sensitive to a MIDI pitch change of 1. Possible future work is to conduct research into converting a MIDI pitch byte to frequency mapping that ensures MIDI pitch changes convert to a frequency pitch change greater than 200 cents, such that it is a noticeable change.
7. Currently, there is little research into coupling the lyrics of music with music visualization. Participants throughout the study stated that it their experience would be more enjoyable if lyrics were provided to them.

6 Appendix

6.1 InstrumentReciever.java Source Code

```java
package player.receivers;

import java.util.concurrent.ConcurrentLinkedQueue;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.atomic.AtomicBoolean;

import javax.sound.midi.MidiMessage;
import javax.sound.midi.Receiver;
import javax.swing.JOptionPane;

import player.messages.OpenGLMessageTonal;
import processors.TonalProcessor;

import controller.Controller;

/**
 * This class acts as the receiver for all MIDI messages. However,
 * this only processes note on/off events for all channels but the
 * beat channel.
 * <p>
 * MIDI messages are described here:
 * http://www.midi.org/techspecs/midimessages.php
 * <p>
 * Each receiver connected to the java sequencer receives the MIDI messages
 * through the {@link #send(MidiMessage, long)} method. This method runs
 * on the sound thread, therefore any processing within this method will
 * slow down the sound thread and therefore cause bugs. Therefore this class
 * is also a thread.
 * <p>
 * As {@link #send(MidiMessage, long)} receives the messages, they are automatically
 * added into {@link ConcurrentLinkedQueue}, which allows for more than one thread to
 * safely add and remove from the queue. Therefore the java sound thread can add all
 * messages to the concurrent queue and then the {@link #run()} thread can dequeue
 * messages and process them without slowing the sequencer thread.
 * <p>
 * @author Michael Pouris
 *
 */
public class InstrumentReceiver extends Thread implements Receiver
{
	private Controller controller;
	private TonalProcessor tonalProcessor;
	private AtomicBoolean playInstuments;
	private LinkedBlockingQueue<MidiMessage> handOffQueue;

	public InstrumentReceiver( Controller controller )
	{
		this.controller = controller;
		this.tonalProcessor = new TonalProcessor();
		this.playInstuments = new AtomicBoolean(true);
		this.handOffQueue = new LinkedBlockingQueue<MidiMessage>();
	}

	public void close()
	{
```

```
}

/**
 * This method runs on the java sound thread, therefore it
 * passes all messages into queue, which then allows another
 * thread to safely access and process the messages.
 * See {@link ConcurrentLinkedQueue} for more details.
 */
public void send(MidiMessage message, long timeStamp)
{
	try
	{
		this.handOffQueue.put(message);
	}
	catch (InterruptedException e)
	{
		JOptionPane.showMessageDialog(null, "InstrumentReceiver::send.
			Could not place " +
				"MidiMessage into LinkedBlockingQueue due to an
					interrupt exception." );
		System.exit(0);
	}
}

/**
 * This method is the thread that dequeues the {@link #handOffQueue}, and
 * processes the MIDI messages without doing any processing on the
 * sound thread.
 * <p>
 * This is a separate thread that is always running and waits for a message to
 * be  added into the queue, therefore not wasting CPU time.
 */
public void run()
{
	int channel;
	MidiMessage message;// = this.test.take();
	byte[] m;

	while(true)
	{
		try {
			message = this.handOffQueue.take();
			if ( message.getStatus() > 127 && message.getStatus() <
				144 && (message.getStatus() - 128) != 9 )
			{
				channel = message.getStatus() - 128;
				m = message.getMessage();

				OpenGLMessageTonal tonalMessage =
					this.tonalProcessor.processNote(
						m[1] & 0xff, 0, channel );
```

```
                                this.controller.getGUI().
                                        getVisualizer().concurrentMessageQueue.get(
                                                channel ).add( tonalMessage );
                                }
                                else if(message.getStatus() > 143 && message.getStatus() <
                                        160 && (message.getStatus() - 144) != 9
                                                && this.playInstuments.get() )
                                {
                                        channel = message.getStatus() - 144;
                                        m = message.getMessage();

                                        OpenGLMessageTonal tonalMessage =
                                                this.tonalProcessor.processNote
                                                        ( m[1] & 0xff, m[2] & 0xff, channel );

                                        this.controller.getGUI().
                                                getVisualizer().concurrentMessageQueue.get(
                                                        channel ).add( tonalMessage );
                                }
                        }
                        catch (InterruptedException e)
                        {
                                JOptionPane.showMessageDialog(null,
                                        "InstrumentReceiver::run. Could not retrieve " +
                                                "MidiMessage from LinkedBlockingQueue due to
                                                        an interrupt exception." );

                                System.exit(0);
                        }
                }
        }

        public AtomicBoolean getIsPlayingInstruments()
        {
                return this.playInstuments;
        }
}
```

6.2 BeatReceiver.java Source Code

```java
package player.receivers;

import java.util.concurrent.ConcurrentLinkedQueue;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.atomic.AtomicBoolean;

import javax.sound.midi.MidiMessage;
import javax.sound.midi.Receiver;
import javax.swing.JOptionPane;

import player.messages.OpenGLMessageBeat;
import processors.BeatProcessor;
import controller.Controller;

/**
 * This class receives all MIDI notes played; however, it only processes
 * note on events for the beat channel. The beat channel is 10 or 9.
 * It is 10 when referring to the channels as 1 to 16. It is 9, when
 * referring to the channels as 0 to 15.
 * <p>
 * Here is the reference to the MIDI standard.
 * http://www.midi.org/techspecs/midimessages.php
 * <p>
 * Each receiver connected to the java sequencer receives the MIDI messages
 * through the {@link #send(MidiMessage, long)} method. This method runs
 * on the sound thread, therefore any processing within this method will
 * slow down the sound thread and therefore cause bugs. Therefore this class
 * is also a thread.
 * <p>
 * The {@link #send(MidiMessage, long)} method adds every {@link MidiMessage} into
 * a {@link ConcurrentLinkedQueue}. This allows the other Thread, the processing
 * thread, to take over.
 * <p>
 * As {@link #send(MidiMessage, long)} receives the messages, they are automatically
 * added into {@link ConcurrentLinkedQueue}, which allows for more than one thread to
 * safely add and remove from the queue. Therefore the java sound thread can add all
 * messages to the concurrent queue and then the {@link #run()} thread can dequeue
 * messages and process them without slowing the sequencer thread.
 * <p>
 * @author Michael Pouris
 *
 */

public class BeatReceiver extends Thread implements Receiver
{
	private Controller controller;
	private BeatProcessor beatProcessor;
	private AtomicBoolean playBeats;
	private LinkedBlockingQueue<MidiMessage> handOffQueue;

	/**
	 * Initialises the object.
	 * <p>
	 * @param controller
	 */
```

```
public BeatReceiver( Controller controller )
{
	this.controller = controller;
	this.beatProcessor = new BeatProcessor();
	this.playBeats = new AtomicBoolean(true);
	this.handOffQueue = new LinkedBlockingQueue<MidiMessage>();
}

public void close()
{
}

/**
 * This method receives a MIDI message in real time and only
 * processes the note on messages sent to channel 9.
 * <p>
 * This method runs on the java sound thread, therefore it
 * passes all messages into queue, which then allows another
 * thread to safely access and process the messages.
 * See {@link ConcurrentLinkedQueue} for more details.
 */
public void send(MidiMessage message, long timeStamp)
{
	try
	{
		this.handOffQueue.put(message);
	}
	catch (InterruptedException e)
	{
		JOptionPane.showMessageDialog(null, "BeatReceiver::send. Could
			not place " +"MidiMessage into LinkedBlockingQueue due to
				an interrupt exception." );
		System.exit(0);
	}
}

/**
 * This method is the thread that dequeues the {@link #handOffQueue}, and
 * processes the MIDI messages without doing any processing on the sound
 * thread. <p>
 * This is a separate thread that is always running and waits for a message to
 * be added into the queue, therefore not wasting CPU time.
 */
public void run()
{
	MidiMessage message;
	byte[] m;

	while( true )
	{
		try
		{
			message = handOffQueue.take();
```

```
                        if ( this.playBeats.get() && message.getStatus() > 143 &&
                                message.getStatus() < 160 &&
                                 (message.getStatus() - 144) == 9 )
                        {
                                m = message.getMessage();
                                OpenGLMessageBeat beat =
                                        beatProcessor.processBeat(m[1] & 0xff,
                                                m[2] & 0xff);

                                this.controller.getGUI().getVisualizer().
                                        concurrentMessageQueue.get(9).add(beat);
                        }
                }
                catch (InterruptedException e)
                {
                        JOptionPane.showMessageDialog(null,
                                "BeatReceiver::run. Could not retrieve " +
                                        "MidiMessage from LinkedBlockingQueue due to
                                                an interrupt exception." );
                        System.exit(0);
                }
        }
    }

    /**
     * Returns the AtomicBoolean that allows for multiple threads to enable
     * and disable the beats from playing into the visualzer.
     * @return
     */
    public AtomicBoolean getIsPlayingBeats()
    {
        return this.playBeats;
    }
}
```

6.3 PitchReceiver.java Source Code

```java
package player.receivers;

import java.util.concurrent.ConcurrentLinkedQueue;
import java.util.concurrent.LinkedBlockingQueue;

import javax.sound.midi.MidiMessage;
import javax.sound.midi.Receiver;
import javax.swing.JOptionPane;

import player.messages.OpenGLMessagePitchChange;

import controller.Controller;

/**
 * This class receives all incoming MIDI messages; however, it only
 * processes pitch bend messages. Pitch bend messages have statuses
 * from 224 to 239. The MIDI messages are explained here:
 * http://www.midi.org/techspecs/midimessages.php
 * <p>
 * Each receiver connected to the java sequencer receives the MIDI messages
 * through the {@link #send(MidiMessage, long)} method. This method runs
 * on the sound thread, therefore any processing within this method will
 * slow down the sound thread and therefore cause bugs. Therefore this class
 * is also a thread.
 * <p>
 * As {@link #send(MidiMessage, long)} receives the messages, they are automatically
 * added into {@link ConcurrentLinkedQueue}, which allows for more than one thread to
 * safely add and remove from the queue. Therefore the java sound thread can add all
 * messages to the concurrent queue and then the {@link #run()} thread can dequeue
 * messages and process them without slowing the sequencer thread.
 * <p>
 * @author Michael Pouris
 *
 */
public class PitchReceiver extends Thread implements Receiver
{
	public final static int MINVALUE = 0x0000;
	public final static int MAXVALUE = 0x3fff;

	private Controller controller;
	private int[] initialPitchSettings;
	private int[] rangeOfPitchValues;
	private LinkedBlockingQueue<MidiMessage> handOffQueue;

	public PitchReceiver( Controller controller )
	{
		this.controller = controller;
		this.initialPitchSettings = new int[16];
		this.rangeOfPitchValues = new int[16];
		this.handOffQueue = new LinkedBlockingQueue<MidiMessage>();
	}

	public void close()
	{
	}
```

```
/**
 * This method runs on the java sound thread, therefore it
 * passes all messages into queue, which then allows another
 * thread to safely access and process the messages.
 * See {@link ConcurrentLinkedQueue} for more details.
 */
public void send(MidiMessage message, long timeStamp)
{
	try
	{
		this.handOffQueue.put(message);
	}
	catch (InterruptedException e)
	{
		JOptionPane.showMessageDialog(null, "PitchReceiver::send. Could
			not place " +"MidiMessage into LinkedBlockingQueue due to
				an interrupt exception." );
		System.exit(0);
	}
}

/**
 * This method is the thread that dequeues the {@link #handOffQueue}, and
 * processes the MIDI messages without doing any processing on the sound
 * thread.
 */
public void run()
{
	MidiMessage message;
	while( true )
	{
		try
		{
			message = this.handOffQueue.take();
			if( message.getStatus() >= 224 &&
				message.getStatus() <= 239 )
			{
				int channel = message.getStatus() - 224;
				byte[] m = message.getMessage();
				int pitchBend = (m[2] & 0xff) << 7 | (m[1] & 0xff);

				double offset =(((double)pitchBend -
					(double)initialPitchSettings[channel])/
						(double)MAXVALUE)*
							rangeOfPitchValues[channel];

				OpenGLMessagePitchChange pitchChange = new
					OpenGLMessagePitchChange(offset,
						channel,rangeOfPitchValues[channel]);

			    if( Math.abs(pitchBend -
					(double)initialPitchSettings[channel]) > 0.5)
			    {
```

```
                                        this.controller.getGUI().getVisualizer().
                                              concurrentMessageQueue.get(channel).
                                                    add(pitchChange);
                                  }
                            }


                        }
                        catch (InterruptedException e)
                        {
                              JOptionPane.showMessageDialog(null, "PitchReceiver::run.
                                    Could not retrieve " +
                                          "MidiMessage from LinkedBlockingQueue due to
                                                an interrupt exception." );
                              System.exit(0);
                        }
                  }
            }

            /**
             * This class handles pitch change messages. Please see the MIDI standard and
             * {@link MidiMessage} for more explanation. However, in order to process
             * these messages, the initial values must be known and the range of values
             * must be known as well. This must be set.
             * <p>
             * @param initialPitchSettings
             * @param rangeOfPitchValues
             */
            public void setPitchData(int[] initialPitchSettings,int[] rangeOfPitchValues)
            {
                  this.initialPitchSettings = initialPitchSettings;
                  this.rangeOfPitchValues = rangeOfPitchValues;
            }
}
```

6.4 ConcurrentVisualizer.java Source Code

```
package visualizer;
import java.awt.Color;
import java.awt.Component;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionListener;
import java.awt.event.MouseWheelEvent;
import java.awt.event.MouseWheelListener;


import javax.media.opengl.GL;
import javax.media.opengl.GL2;
import javax.media.opengl.GL2ES1;
import javax.media.opengl.GLAutoDrawable;
import javax.media.opengl.GLEventListener;

import javax.media.opengl.fixedfunc.GLLightingFunc;
import javax.media.opengl.fixedfunc.GLMatrixFunc;
import javax.media.opengl.glu.GLU;



import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ConcurrentLinkedQueue;
import java.util.concurrent.atomic.AtomicBoolean;

import player.messages.OpenGLMessage;
import player.messages.OpenGLMessageBeat;
import player.messages.OpenGLMessagePitchChange;
import player.messages.OpenGLMessageTonal;
import visualizer.camera.Camera;


public class ConcurrentVisualizer extends Thread implements GLEventListener,
	MouseMotionListener, KeyListener, MouseWheelListener
{
	public static final int MAX_CHANNELS = 16;
	public static final int MAX_BEAT_PIPES = 5;
	public static final int MAX_PIPES_PER_CHANNEL = 2;
	public static final int FLOATS_USED_PER_3D_POINT = 3;
	public static final int FLOATS_USED_PER_COLOUR = 4;
	public static final float MIN_SIZE_FOR_RADIUS = 1;

	private GLU glu;
	private Camera camera;

/	/Used to assist the mouse
	private float prevX, prevY;

	//private FPSTimer timer;

	private ConcurrentPipe[][] pipes;
	private Beat[] beats;
	private ConcurrentHashMap<Integer, Boolean> pipesToUse;
```

```
private float[][] beatColours =
	{{1,0,0,1},{0,1,0,1},{0,0,1,1},{0,1,1,1},{1,1,0,1}};
private Color colours[] = {
		new Color(255, 0, 0),  new Color(255, 72, 0), new Color(255, 145,
			0),new Color(255, 217, 0),
		new Color(217, 255, 0),new Color(145, 255, 0),new Color(72, 255,
			0) ,new Color(0, 255, 217),
		new Color(0, 217, 255),new Color(0, 145, 255),new Color(0, 72,
			255) ,new Color(0, 0, 255),
		new Color(145, 0, 255),new Color(217, 0, 255),new Color(255, 0,
			217),new Color(255, 0, 145)
		};

//Setting up the lights: the light position
private float lightPosition0[] = {30,0,-30,1};
 private float[] lightAmbient0 = {0.1f,0.1f,0.1f,1.0f};
 private float[] lightDiffuse0 = {1.0f,1.0f,1.0f,1.0f};

private float lightEmissiveMaterial[] = {0.1f, 0.1f, 0.1f, 1.0f};
 private float brassAmbientMaterial[] = {0.33f, 0.22f, 0.03f, 1.0f};
private float brassDiffuseMaterial[] = {0.78f, 0.57f, 0.11f, 1.0f};
 private float brassSpecularMaterial[] = {1,1,1,1};

private int[][] lastPitchInChannelAndPipe;
private int[][] lastVolumeInChannelAndPipe;
private float[] pitchBends;
private float[][][] lastCoorindates;
private float[][] lastDimensions;

public ConcurrentHashMap<Integer, ConcurrentLinkedQueue<OpenGLMessage>>
	concurrentMessageQueue;

private AtomicBoolean[] activatedBeat;

public ConcurrentVisualizer()
{
	this.concurrentMessageQueue = new ConcurrentHashMap<Integer,
		ConcurrentLinkedQueue<OpenGLMessage>>();

	this.activatedBeat = new AtomicBoolean[MAX_BEAT_PIPES];
	for( int i = 0; i < MAX_BEAT_PIPES; i++  )
	{
		this.activatedBeat[i] = new AtomicBoolean(false);
	}

	this.lastPitchInChannelAndPipe = new
		int[MAX_CHANNELS][MAX_PIPES_PER_CHANNEL];

	this.lastVolumeInChannelAndPipe = new
		int[MAX_CHANNELS][MAX_PIPES_PER_CHANNEL];

	this.lastCoorindates = new
		float[MAX_CHANNELS][MAX_PIPES_PER_CHANNEL][FLOATS_USED_PER_3D_POI
	NT];
	this.lastDimensions = new float[MAX_CHANNELS][MAX_PIPES_PER_CHANNEL];
```

```
        this.pitchBends = new float[MAX_CHANNELS];

        for( int i = 0; i < MAX_CHANNELS; i++ )
        {
                for( int j = 0; j < MAX_PIPES_PER_CHANNEL; j++ )
                {
                        this.lastPitchInChannelAndPipe[i][j] = 0;
                        this.lastVolumeInChannelAndPipe[i][j] = 0;
                        this.lastDimensions[i][j] = 0;

                        for( int k = 0; k < FLOATS_USED_PER_3D_POINT; k++ )
                        {
                                this.lastCoorindates[i][j][k] = 0;
                        }
                }
                pitchBends[i] = 0;
        }

        for( int i = 0; i < MAX_CHANNELS; i++ )
        {
                this.concurrentMessageQueue.put
                        (i, new ConcurrentLinkedQueue<OpenGLMessage>() );
        }

        this.pipesToUse = new ConcurrentHashMap<Integer, Boolean>(2, 0.9f, 2);
}

public void display(GLAutoDrawable drawable)
{
        final GL2 gl = drawable.getGL().getGL2();
        gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
        gl.glLoadIdentity();

        camera.update();
        gl.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_EMISSION,
                lightEmissiveMaterial, 0);

        gl.glLightfv(GLLightingFunc.GL_LIGHT0, GLLightingFunc.GL_POSITION,
        lightPosition0, 0);

    for( Integer i : this.pipesToUse.keySet() )
    {
        for( int j = 0; j < MAX_PIPES_PER_CHANNEL; j++ )
        {
                pipes[i][j].draw(drawable);
        }
    }

    for( int i = 0; i < MAX_BEAT_PIPES; i++ )
    {
        beats[i].draw(drawable, activatedBeat[i].get());
        gl.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_EMISSION,
                lightEmissiveMaterial, 0);
```

```
}
   }

   /**
    * Considering the JOGL system runs the OpenGL thread on the EventDispatch
    * thread, any calculations will slow the whole system down. The solution to
    * this problem is the same solution used with the receivers, where a thread
    * is created that handles the processing therefore to keep the GUI speed.
    *
    * @see player.receivers.BeatReceiver
    * @see player.receivers.ControlReceiver
    * @see player.receivers.InstrumentReceiver
    * @see player.receivers.PitchReceiver
    */
   public void run()
   {
          while(true)
          {
                 for( int i = 0; i < 10; i++ )
                 {
                        for( int j = 0; j < MAX_CHANNELS; j++ )
                        {
                               ConcurrentLinkedQueue<OpenGLMessage> queue =
                                      concurrentMessageQueue.get(j);

                               OpenGLMessage message = queue.poll();
                               if(message == null )
                               {
                                      continue;
                               }
                               if( message.getMessage() == OpenGLMessage.NOTEON  ||
                                      message.getMessage()== OpenGLMessage.NOTEOFF)
                               {
                                  processTones((OpenGLMessageTonal)message );
                               }
                               else if( message.getMessage() ==
                                      OpenGLMessage.BEAT )
                               {
                                      activatedBeat[((OpenGLMessageBeat)message).ge
                                             tPipe()].set(true);
                               }
                               else if( message.getMessage() ==
                                      OpenGLMessage.PITCHCHANGES )
                               {
                                      processPitchChanges(
                                      (OpenGLMessagePitchChange)message );
                               }
                        }
                 }
                 try
                 {
                        Thread.sleep(100);
                 }
                 catch (InterruptedException e)
                 {
```

```
                    e.printStackTrace();
                }
        }
}

/**
 * Processes all tonal messages {@link OpenGLMessageTonal} and creates
 * the proper animations.
 * <p>
 * @param message the message to process
 */
private void processTones( OpenGLMessageTonal message )
{
        int channel = message.getChannel();
        int pipe = message.getPipe();
        int note = message.getNote();
        int velocity = message.getVolume();
        int noteDifference = 0;
        float alpha;
        float loomingPosition = 0;
        float relativeNoteOnGrid;
        float xAndY;
        float[] coordinates = new float[3];

        //The pipe that we are altering
        ConcurrentPipe p = pipes[channel][pipe];

        //Calculate the alpha for the new face
        //alpha = 0.003937f*velocity +0.5f;
        alpha = (0.5f/127f)*velocity + 0.5f;

        //Create a new position on the screen
        relativeNoteOnGrid = 5.5f * (note - 64);

        //Alter the positioning on the screen based
        noteDifference = note - lastPitchInChannelAndPipe[channel][pipe];
        if( Math.abs(noteDifference) >= 4 && Math.abs(noteDifference) <=15 )
        {
                loomingPosition =(float)( 1.5*( Math.abs(noteDifference) ) +
                        0.5*( velocity -
                                lastVolumeInChannelAndPipe[channel][pipe]) );

                loomingPosition = loomingPosition / 4;
        }

        //Save the last note and velocity it was played at. This will aide in
        //figuring out how to position the new face
        lastPitchInChannelAndPipe[channel][pipe] = note;
        lastVolumeInChannelAndPipe[channel][pipe] = velocity;

        if( message.getMessage() == OpenGLMessageTonal.NOTEON )
        {
                //Calulate the size of the new face
```

```
                coordinates =  p.getInitialPlacement();
                coordinates[0] += loomingPosition;
                coordinates[1] = relativeNoteOnGrid;
                coordinates[2] = 0;

                lastCoorindates[channel][pipe] = coordinates;
                lastDimensions[channel][pipe] = xAndY;

                float newFace[][] =  p.createNewFace(xAndY, coordinates[0],
                        coordinates[1], 0);

                //Must set the firstFaceData in order for the pipe to show the
                //animations.
                p.getPositionAnimationList().add(newFace);
                p.getAlphaAnimationList().add(alpha);
                p.setFirstFaceData(xAndY);
        }
        else if( message.getMessage() == OpenGLMessageTonal.NOTEOFF )
        {
                coordinates =  p.getInitialPlacement();
                coordinates[1] = lastCoorindates[channel][pipe][1];
                coordinates[2] = 0;

                lastDimensions[channel][pipe] = 0;
                lastCoorindates[channel][pipe] = coordinates;

                float newFace[][] =  p.createNewFace(0, coordinates[0],
                        coordinates[1], 0);

                //Must set the firstFaceData in order for the pipe to show the
                //animations.
                p.getPositionAnimationList().add(newFace);
                p.getAlphaAnimationList().add(alpha);
                p.setFirstFaceData(0);
        }
/**
 * This method processes the pitch changes from the Midi Sequencer.
 * When a midi pitch change event is passed to the visualizer, it
 * is processed, which tells the channel how to act. Then the pitch
 * change needs to be applied to all the pipes in the channel. Therefore
 * the method works in 2 parts:
 * 1. Calculates the pitch change value. (At this point the change event
 *           is not applied to any pipes)
 *
 * 2. Apply the pitch changes to the channel's pipe.
 *           This is done in the same way was a NoteOn/Off event
 *           is applied to a pipe.
 *
 * @param pitchMessage: OpenGLMEssagePitchChange sent from the
 *           MidiNoteReceiver
 */
private void processPitchChanges( OpenGLMessagePitchChange pitchMessage )
{
        //1. Set the pitch bends for that one channel
```

```
                double scale =
                        (double)15/(double)(pitchMessage.getRangeOfPitchValues());

                double pitchWheelValue =  scale * pitchMessage.getOffset();
                pitchBends[channel] = (float)(pitchWheelValue);//*3.5);

                //2. Apply same note for the channel affected
                ConcurrentPipe p;
                float[][] newFace;
                for( int i = 0; i < MAX_PIPES_PER_CHANNEL; i++)
                {
                        p = pipes[channel][i];

                        //Take the note the was last played for the channel's pipe, then
                        //apply the pitch bend to the y axis
                        newFace = p.createNewFace(lastDimensions[channel][i],
                                lastCoorindates[channel][i][0],
                                        lastCoorindates[channel][i][1]+pitchBends[channel]
                                         , 0);

                        //Add the new face to the animation list
                        p.getPositionAnimationList().add(newFace);
                        lastCoorindates[channel][i][1] += pitchBends[channel];
                        p.setFirstFaceData(lastDimensions[channel][i]);
                        lastCoorindates[channel][i][1] -= pitchBends[channel];
                }
        }

/**
         * This method is called first when the visualizer OpenGL is started.
         * Set up opengl's state machine. These are enabled/disabled here because
         * they are staying constant throughout the life of the program. Therefore
         * the overhead is needed only once. The meaning of the parameters and the
         * functions are all normal and basic openGL calls, they can be looked up in
         * the JOGL API or the OpenGL api.
         */
        public void init(GLAutoDrawable drawable)
        {
                GL2 gl = drawable.getGL().getGL2();
                gl.glShadeModel(GLLightingFunc.GL_SMOOTH);
                gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
                gl.glClearDepth(1.0f);
                gl.glEnable(GL.GL_DEPTH_TEST);
                gl.glDepthFunc(GL.GL_LEQUAL);
                gl.glHint(GL2ES1.GL_PERSPECTIVE_CORRECTION_HINT, GL.GL_FASTEST);
                gl.glDisable(GL.GL_LINE_SMOOTH);
                gl.glEnable(GL2.GL_TEXTURE_2D);
                gl.glEnable(GL2.GL_NORMALIZE);
                gl.glEnable (GL.GL_BLEND);
                gl.glEnable(GL2.GL_LIGHTING);
                gl.glEnable(GL2.GL_COLOR_MATERIAL);
                gl.glBlendFunc (GL.GL_SRC_ALPHA, GL.GL_ONE_MINUS_SRC_ALPHA);
                gl.glEnable(GL.GL_CULL_FACE);
            gl.glCullFace(GL.GL_BACK);
```

```
        gl.glMaterialfv(GL.GL_FRONT_AND_BACK, GL2.GL_AMBIENT,
        brassAmbientMaterial,0);

      gl.glMaterialfv(GL.GL_FRONT_AND_BACK, GL2.GL_DIFFUSE,
        brassDiffuseMaterial,0);

      gl.glMaterialfv(GL.GL_FRONT_AND_BACK, GL2.GL_SPECULAR,
        brassSpecularMaterial,0);

      gl.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_EMISSION,
        lightEmissiveMaterial, 0);

      gl.glMaterialf(GL.GL_FRONT_AND_BACK, GL2.GL_SHININESS, 27);

      gl.glEnableClientState(GL2.GL_VERTEX_ARRAY);
      gl.glEnableClientState(GL2.GL_COLOR_ARRAY);
      gl.glEnableClientState(GL2.GL_NORMAL_ARRAY);

        gl.glLightfv(GL2.GL_LIGHT0, GL2.GL_AMBIENT, lightAmbient0,0);
        gl.glLightfv(GL2.GL_LIGHT0, GL2.GL_DIFFUSE, lightDiffuse0,0);
        gl.glEnable(GL2.GL_LIGHT0);

        glu = new GLU();
        camera = new Camera(0, 80,500,-50,0,0);

((Component) drawable).addKeyListener(this);

        //Create the pipes and position them properly with their appropriate
        //color
        int amountOfSections = 200;
        float[] initialColour = {1,0,0,1f};
        float[] initialPosition = {0,0,0};
        pipes = new ConcurrentPipe[MAX_CHANNELS][MAX_PIPES_PER_CHANNEL];

        for( int i = 0; i < MAX_CHANNELS; i++ )
        {
              for( int j = 0; j < MAX_PIPES_PER_CHANNEL; j++ )
              {
                    initialColour[0] = (float) (colours[i].getRed() / 255.0);

                    initialColour[1] = (float) (colours[i].getGreen() /
                          255.0);

                    initialColour[2] = (float) (colours[i].getBlue() / 255.0);

                    pipes[i][j] = new ConcurrentPipe(0,0, amountOfSections,
                          initialColour, initialPosition);
                    pipes[i][j].createPipe(drawable);
                    initialPosition[1] -= 7;
              }
              initialPosition[1] = 0;
              initialPosition[0] = initialPosition[0]-25;
        }
```

```
initialPosition[1] = 0;
	initialPosition[2] = 0;

	int positions[] = {-130,-60,-240,0,-200};
	int sizes[] = {250,33,15,15,33};
	int iSizes[] = {235,20,11,11,22};
	beats = new Beat[MAX_BEAT_PIPES];
	for( int i = 0; i < MAX_BEAT_PIPES; i++ )
	{
		initialPosition[0] = positions[i];
		beats[i] = new Beat(initialPosition, beatColours[i],
		amountOfSections, sizes[i],iSizes[i]);
	}
}

/**
 * This method is called right after the Init() method, when the screen is
 * created. Therefore set up the projection and model/view matrix. Set the
 * model/view matrix  last because the draw method is called right after. When
 * the Visualizer is  started, the methods are called in this order:
 *		1. init()
 *		2. reshape()
 *		3. draw()
 * The methods are apart of GLEventListener in JOGL
 */
public void reshape(GLAutoDrawable drawable, int x, int y, int width,int
	height)
{
	GL2 gl = drawable.getGL().getGL2();
	if (height <= 0)
	{
		height = 1;
	}
	float h = (float) width / (float) height;

	//Disables vertical sync and therefore increases the speed
	gl.setSwapInterval(0);

	//Set the Projection matrix. Please note that this method is called
	//right after the
	//the init method. Therefore I just place this in the reshape function
	gl.glMatrixMode(GLMatrixFunc.GL_PROJECTION);
	gl.glLoadIdentity();
	glu.gluPerspective(50.0f, h, 1.0, 3000.0);

	//Set the model view matrix
	gl.glMatrixMode(GLMatrixFunc.GL_MODELVIEW);
	gl.glLoadIdentity();
}

/**
 * Sets the channels to animate.
 *
 * @param pipesToUse
```

```
public void setChannelsUsed(int[] pipesToUse)
{
	for( int i = 0; i < 16; i++ )
	{
		this.pipesToUse.remove(i);
	}
	for( int i = 0; i < pipesToUse.length ; i++ )
	{
		this.pipesToUse.put(pipesToUse[i], true);
	}

}

/**
 * This is the next part in the mouse movement that
 * actually moves the camera around.
 * When the mouse is clicked then dragged the camera is
 * moved around.
 */
public void mouseDragged(MouseEvent e)
{
	camera.positionCamera(e.getX()-prevX, e.getY()-prevY);
	prevX = e.getX();
	prevY = e.getY();
}

/**
 * This method is used to help move the camera around
 * the visualization in a circular motion. The method
 * records the position of the mouse on the window every time
 * it moves.
 */
public void mouseMoved(MouseEvent e)
{
	prevX = e.getX();
	prevY = e.getY();
}

/**
 * Handles the changes to the vizualization when keys are pressed.
 * So far it handles: When the camera is moved forward and
 * and the camera is moved back.
 */
public void keyPressed(KeyEvent e)
{
  if( e.getKeyCode() == KeyEvent.VK_UP  )
  {
	camera.moveCameraForward();
  }
  if( e.getKeyCode() == KeyEvent.VK_DOWN	)
  {
	camera.moveCameraBackward();
  }

}
```

```
/**
 * Resets the animation
 */
public void resetVisualizer()
{
	for( int i = 0; i < MAX_CHANNELS; i++ )
	{
		for( int j = 0; j < MAX_PIPES_PER_CHANNEL; j++ )
		{
			pipes[i][j].resetPipeAnimation();
		}
	}
}

/**
 * Whenever the mouse wheel on a physical mouse is rolled,
 * the camera will zoom in or zoom out. That depends
 * on which way the mouse is rolled.
 */
public void mouseWheelMoved(MouseWheelEvent e)
{
	camera.setZoom( 10 * e.getWheelRotation() );
}

public void dispose(GLAutoDrawable drawable){}
public void keyReleased(KeyEvent e){}
public void keyTyped(KeyEvent e){}
```

6.5 Beat.java Source Code

```java
package visualizer;

import javax.media.opengl.GL2;
import javax.media.opengl.GLAutoDrawable;
import javax.media.opengl.glu.GLU;
import javax.media.opengl.glu.GLUquadric;

import com.jogamp.opengl.util.gl2.GLUT;

public class Beat
{
	private static final int AMOUNT_BETWEEN_FACES = 4;
	private static final int SLICES = 15;

	private final GLU glu;
	private final GLUquadric quadric;
	private final int amountOfFaces;
	private final float[] initialPlacement;
	private final float[] initialColour;
	private final int maxLengthOfBeat;
	private int radius;
	private int innerRadius;
	private int position;
	private GLUT glut;

	/**
	 * Creates a new beat object. The object stores the initialColour,
	 * initialPlacement and amountOfFaces for its whole life; these variables are
	 * final and CANNOT be changed.
	 * @param initialPlacement: Stores the X,Y and Z position of the pipe. It is
	 * good to note that the Z position is not used by the object because it is
	 * used to animate the pipe.
	 * @param initialColour: Stores the initial RGBA as floats (0 -> 1).
	 * @param amountOfFaces: This is specifically used with the pipes. Because the
	 * pipes have a set distance between their faces, it must be kept track of in
	 * animate the beat objects in the same amount of time.
	 * @param radius: The radius of the beat.
	 */
	public Beat( float[] initialPlacement, float[] initialColour, int
		amountOfFaces, int radius, int innerRadius )
	{
		this.glut = new GLUT();
		this.glu = new GLU();
		this.radius = radius;
		this.innerRadius = innerRadius;
		this.quadric = this.glu.gluNewQuadric();
		this.amountOfFaces = amountOfFaces;
		this.initialPlacement = initialPlacement.clone();
		this.initialColour = initialColour.clone();
		this.maxLengthOfBeat = AMOUNT_BETWEEN_FACES * this.amountOfFaces;
		this.position = maxLengthOfBeat;
		this.glu.gluQuadricNormals(this.quadric,GLU.GLU_SMOOTH);
	}

	/**
```

```
 * Draws the beat on the screen. When the object is
 * initially created, the beat is not animating. When
 * a boolean value of true is sent, the beat is animated
 * from the 0th position to the nth position. If a draw = true
 * is passed, then the beat is forced back to the beginning to
 * animate. This creates a snapping image like a real drum.
 *
 * @param drawable: The openGL context
 * @param draw: Is a boolean that alerts the beat that a new
 *            beat signal was sent to the object, which in turn
 *            animates the beat.
 */

public void draw(GLAutoDrawable drawable, boolean draw)
{
	GL2 gl = drawable.getGL().getGL2();
	//Start animating from the beginning (0th position)
	if( draw )
	{
		position = 0;
	}
	//The beat is finished animating because it is at the end
	//therefore do not draw the beat. Do this by returning.
	if( position >= maxLengthOfBeat )
	{
		return;
	}

	//Animate the beat
	gl.glPushMatrix();
		animate(drawable);
	gl.glPopMatrix();

	//Move the beat back. This is what achieves the animation effect.
	position = position + AMOUNT_BETWEEN_FACES;
}

/**
 * Sets the colour, position and draws the physical beat.
 * @param drawable
 */
private void animate(GLAutoDrawable drawable)
{
	GL2 gl = drawable.getGL().getGL2();

	float currentAlpha = 1f - (position / (float) maxLengthOfBeat);

	//float emissiveLight0[] = {0.1f, 0.1f, 0.1f, 1.0f};
	float emissiveLight1[] = {1f, 0f, 0f, 1.0f};
	if( position <= 25 )
	{
		emissiveLight1[0] = 0.75f*initialColour[0] - (position / (float)
			25);
		emissiveLight1[1] = 0.75f*initialColour[1] - (position / (float)
			25);
```

```
                        emissiveLight1[2] = 0.75f*initialColour[2] - (position / (float)
                              25);
                        gl.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_EMISSION,
                        emissiveLight1, 0);
                  }

                  //Position and draw it
                  gl.glTranslatef(initialPlacement[0], initialPlacement[1], position);
                  gl.glColor4f(initialColour[0],initialColour[1],initialColour[2],
                        currentAlpha);

                  glut.glutSolidTorus(radius-innerRadius, radius, SLICES, 60);
            }
      }
```

6.6 ConcurrentPipe.java

```
package visualizer;

import java.nio.FloatBuffer;
import java.nio.IntBuffer;
import java.util.ArrayList;
import java.util.concurrent.ConcurrentLinkedQueue;
import java.util.concurrent.atomic.AtomicReference;

import javax.media.opengl.GL;
import javax.media.opengl.GL2;
import javax.media.opengl.GLAutoDrawable;
import javax.media.opengl.glu.GLU;
import javax.media.opengl.glu.GLUquadric;

public class ConcurrentPipe
{
	//Amount of verticies per face
	public static final int AMOUNT_OF_VERTS = 6;
	public static final int FLOATS_USED_PER_COLOUR = 4;
	public static final int FLOATS_USED_PER_3D_POINT = 3;
	public static final float INITIAL_RADIUS = 0;
	private static final int AMOUNT_BETWEEN_FACES = 4;
	private static final float PERCENTAGE_OF_PIPE_TO_FADE = 0.75f;

	//Amount of FLOATS that make up the vertex and its information.
	//The information for a vertex is placed as follows:
	//VVVCCCCNNN. Therefore there are 10 floats for the information for a vertex.
	//Each V,C and N are a signal number specified as a float.
	private static final int SIZE_OF_VERTEX_INFORMATION_IN_FLOATS = 10;
	private static final int SIZE_OF_VERTEX_INFORMATION_IN_BYTES =
		SIZE_OF_VERTEX_INFORMATION_IN_FLOATS * (Float.SIZE/Byte.SIZE);
	private static final int START_INDEX_OF_VERTEX = 0;
	private static final int START_INDEX_OF_COLOUR = START_INDEX_OF_VERTEX +
		FLOATS_USED_PER_3D_POINT * (Float.SIZE/Byte.SIZE);
	private static final int START_INDEX_OF_NORMAL = START_INDEX_OF_COLOUR +
		FLOATS_USED_PER_COLOUR * (Float.SIZE/Byte.SIZE);

	//The amount of sections the pipe has. Note: 1 section has 2 faces.
	private final int amountOfSections;

	//The amount of faces the pipe has. Note: The amount of faces
	//is the amountOfSections + 1
	private final int amountOfFaces;

	//To make a pipe with 3 sides (AKA triangle), we need 4 verticies.
	//Therefore, the amount of sides we have is: AMOUNT_OF_VERTS - 1
	private final int amountOfSides;

	//These are used for the placement of the pipes and random operations
	//which are used for testing the object.
	private final float[] initialPlacement;
	private final float[] initialColor;

	//Holds the order that the verticies are drawn in
	private ArrayList<Integer> indicies;
```

```
//Pointer to the pipe data in graphics memory. This must be bound
//before use. Usage is altering and drawing. The same is for the index
//pointer. It holds the pointer to the data in graphics memory.
//The data specifies the order to draw the verticies in.
private int pipePointer;
private int indexPointer;

private boolean isCreated = false;

//private float[][] lastFace;
private AtomicReference<float[][]> lastFace;
private ConcurrentLinkedQueue<float[][]> positionAnimationQueue;

//private float lastAlpha;
private AtomicReference<Float> lastAlpha;
private ConcurrentLinkedQueue<Float> alphaAnimationQueue;

//private float currentRadius;
private AtomicReference<Float> currentRadius;
private ConcurrentLinkedQueue<Float> radiusAnimationQueue;
private int amountOfFacesNotDrawn;

//Identifies the pipe's channel and the part of the channel it belongs to
private final int channel;
private final int pipe;

//Used for the ball at the end
private GLU glu;
private GLUquadric quadric;

//The pipe is faded out at the back. Use this to store where the fading starts
private final int minValueAlphaFading;

/**
 * Once the pipe has been initialized, this method must be called right after
 * in order to create the amount of memory needed in the graphics card. After
 * the proper amount of memory was allocated by the pipe, the pipe is built
 * and put into memory. Only after this method has finished, can the pipe be
 * drawn and animated.
 */
public void createPipe( GLAutoDrawable drawable )
{
	if( !this.isCreated )
	{
		GL2 gl = drawable.getGL().getGL2();

		//Calculate the size (in bytes) that we need for the verticies,
		//colors and normals
		int byteSizeForVerticies = amountOfFaces * AMOUNT_OF_VERTS *
			FLOATS_USED_PER_3D_POINT * (Float.SIZE/Byte.SIZE);
		int byteSizeForColors = amountOfFaces * AMOUNT_OF_VERTS *
			FLOATS_USED_PER_COLOUR * (Float.SIZE/Byte.SIZE);
		int byteSizeForNormals = amountOfFaces * AMOUNT_OF_VERTS *
			FLOATS_USED_PER_3D_POINT * (Float.SIZE/Byte.SIZE);
```

```
//Create the buffer, bind it for use, then allocate the amount of
//memory needed.
//The get the data as a float buffer.
int[] pointers = new int[2];
gl.glGenBuffers(1, pointers, 0);
gl.glBindBuffer(GL2.GL_ARRAY_BUFFER, pointers[0]);
gl.glBufferData(GL2.GL_ARRAY_BUFFER, byteSizeForVerticies +
      byteSizeForColors + byteSizeForNormals, null,
            GL2.GL_DYNAMIC_DRAW);
FloatBuffer floatBuffer = gl.glMapBuffer(GL2.GL_ARRAY_BUFFER,
      GL2.GL_WRITE_ONLY).asFloatBuffer();

//Draw the first face.
//dataCounter is used to keep track of the float we are altering
int dataCounter = -1;
float x,y,z,eZ = 0;

for( int j = 0; j < AMOUNT_OF_VERTS; j++ )
{
      x = (float) ( INITIAL_RADIUS *
            Math.sin(2*j*Math.PI/(amountOfSides))) +
                  initialPlacement[0];

      y = (float) ( INITIAL_RADIUS *
            Math.cos(2*j*Math.PI/(amountOfSides))) +
                  initialPlacement[1];

      z = 0 + initialPlacement[2];

      //Setting the vertex
      dataCounter++;
      floatBuffer.put(dataCounter, x);
      dataCounter++;
      floatBuffer.put(dataCounter, y);
      dataCounter++;
      floatBuffer.put(dataCounter, z );

      //Setting the color
      dataCounter++;
      floatBuffer.put(dataCounter, initialColor[0]);
      dataCounter++;
      floatBuffer.put(dataCounter, initialColor[1]);
      dataCounter++;
      floatBuffer.put(dataCounter, initialColor[2]);
      dataCounter++;
      floatBuffer.put(dataCounter, initialColor[3]);

      //The normal calculations and placement in buffer
      x = (float) ( Math.sin(2*j*Math.PI/(amountOfSides)));
      y = (float) ( Math.cos(2*j*Math.PI/(amountOfSides))) ;

      dataCounter++;
      floatBuffer.put(dataCounter, x);
      dataCounter++;
      floatBuffer.put(dataCounter, y);
```

```
        dataCounter++;
        floatBuffer.put(dataCounter, 0);
}

//Make the second face a certain length away.
eZ += AMOUNT_BETWEEN_FACES;

//The first face is done, therefore I have to make a second face
//that is
//one away from the first. Each consecutive face is one away.
for( int i = 0; i < this.amountOfSections; i++ )
{
        for( int j = 0; j < AMOUNT_OF_VERTS; j++ )
        {
                x = (float) ( INITIAL_RADIUS *
                        Math.sin(2*j*Math.PI/(amountOfSides))) +
                                initialPlacement[0];

                y = (float) ( INITIAL_RADIUS *
                        Math.cos(2*j*Math.PI/(amountOfSides))) +
                                initialPlacement[1];

                z = 0  + initialPlacement[2];

                //The Vertex
                dataCounter++;
                floatBuffer.put(dataCounter, x);
                dataCounter++;
                floatBuffer.put(dataCounter, y);
                dataCounter++;
                floatBuffer.put(dataCounter, z + eZ);

                //The Color
                dataCounter++;
                floatBuffer.put(dataCounter, initialColor[0]);
                dataCounter++;
                floatBuffer.put(dataCounter, initialColor[1]);
                dataCounter++;
                floatBuffer.put(dataCounter, initialColor[2]);
                dataCounter++;
                floatBuffer.put(dataCounter, initialColor[3]);

                //The normal calculations and placement in buffer
                x = (float) (Math.sin(2*j*Math.PI/(amountOfSides)));
                y = (float) (Math.cos(2*j*Math.PI/(amountOfSides)));

                dataCounter++;
                floatBuffer.put(dataCounter, x);
                dataCounter++;
                floatBuffer.put(dataCounter, y);
                dataCounter++;
                floatBuffer.put(dataCounter, 0);
        }
        eZ = eZ + AMOUNT_BETWEEN_FACES;
```

```
                }

                lastFace.set(getFace(floatBuffer, 0));

                gl.glUnmapBuffer(GL2.GL_ARRAY_BUFFER);

                //The vertices and their respective color and normal information
                //are now stored.
                //Now we have to store the order at which the verticies need to
                //be drawn.
                //For example, just say I have 2 faces. I Stored the information
                //as:
                //Face 0: 0,1,2,3,4,5 Face 1: 6,7,8,9,10,11. To draw a cylinder,
                //we need to use
                //a triangle strip. Therefore, I cannot draw the verticies from 0
                //to 10,
                //I must draw it 0,6,1,7,2,8,3,9,4,10,5,11
                this.indicies = new ArrayList<Integer>();
                for( int i = 0; i < amountOfFaces - 1; i++ )
                {
                      for( int j = i * AMOUNT_OF_VERTS; j <
                            (i+1)*AMOUNT_OF_VERTS; j++ )
                      {
                            this.indicies.add(j);
                            this.indicies.add(j+AMOUNT_OF_VERTS);
                      }
                }

                //Now Generate the IBO. This explains to the video card how to
                //access the veritices
                gl.glGenBuffers(1, pointers,1);
                gl.glBindBuffer(GL.GL_ELEMENT_ARRAY_BUFFER, pointers[1]);
                gl.glBufferData(GL.GL_ELEMENT_ARRAY_BUFFER,
                      (Integer.SIZE/Byte.SIZE) * indicies.size(), null,
                            GL.GL_STATIC_DRAW);
                IntBuffer intBuff = gl.glMapBuffer(GL.GL_ELEMENT_ARRAY_BUFFER,
                      GL.GL_WRITE_ONLY).asIntBuffer();
                for( int i = 0; i < indicies.size(); i++ )
                {
                      intBuff.put(i, indicies.get(i));
                }
                gl.glUnmapBuffer(GL.GL_ELEMENT_ARRAY_BUFFER);

                //This is the pointers that I will use to access the buffer
                //memory on the graphics card.
                this.pipePointer = pointers[0];
                this.indexPointer = pointers[1];
                this.isCreated = true;
          }
}

/**
 *     Sets the primitives that the pipe object uses. It does not create the
 * memory space
 * in the graphics card and it does not initialize the pipe object in graphics
```

```
* memory.
 * This constructor tells the pipe: 1. How many sections is needed, the color
 * and the placement in the world.
 * To initialize the pipe fully (create the pipe in graphics memory space so
 * it can be drawn), obj.createPipe(...)
 * must be called.
 *
 * @param int amountOfSections: The amount of sections the pipe has. The
 * amount of faces
 * a pipe has is one more than the amount of sections. For example, if a pipe
 * with one section
 * is wanted, then it has 2 face: the beginning face and the end face. For 2
 * sections, there are
 * 3 faces: The beginning face, the middle face and the end face.
 *
 * @param float color: The color of the pipe in RGBA
 *
 * @param float translations: The initial placement of the pipe in the world.
 */
public ConcurrentPipe( int channel, int pipe, int amountOfSections, float[]
color, float[] translations )
{
	//A flag stating that the pipe has not been created in graphics memory
	this.isCreated = false;

	this.channel = channel;
	this.pipe = pipe;

	//Copies the colours and placement of the pipe
	this.initialPlacement = translations.clone();
	this.initialColor = color.clone();

	//Sets the amount of sections wanted, sets the amount of faces
	//as well as the number of sides that is needed.
	this.amountOfSections = amountOfSections;
	this.amountOfFaces = amountOfSections + 1;
	this.amountOfSides = ConcurrentPipe.AMOUNT_OF_VERTS - 1;

	this.positionAnimationQueue = new ConcurrentLinkedQueue<float[][]>();
	this.alphaAnimationQueue = new ConcurrentLinkedQueue<Float>();

	//this.lastFace= new float[AMOUNT_OF_VERTS][FLOATS_USED_PER_3D_POINT];
	this.lastFace = new AtomicReference<float[][]>(new
	float[AMOUNT_OF_VERTS][FLOATS_USED_PER_3D_POINT]);
	//this.lastAlpha = 1f;
	this.lastAlpha = new AtomicReference<Float>(1f);

	this.glu = new GLU();
	this.quadric = this.glu.gluNewQuadric();

	//this.amountOfFacesNotDrawn = 0;
	//this.currentRadius = 0;

	this.minValueAlphaFading = (int) (PERCENTAGE_OF_PIPE_TO_FADE *
	this.amountOfFaces);
```

```
        //this.currentRadius = 0;
        this.currentRadius = new AtomicReference<Float>(0f);
        this.radiusAnimationQueue = new ConcurrentLinkedQueue<Float>();
        this.amountOfFacesNotDrawn = 0;
}

public void draw(GLAutoDrawable drawable)
{
        GL2 gl = drawable.getGL().getGL2();

        //animate(drawable);

        //If the current radius is (essentially 0), then the face drawn
        //will not be displayed, it will look as if the face is not there.
        //This uses more cpu/gpu power than needed. Therefore keep track
        //of the faces that have a radius of 0. If there is a radius
        //not 0, then that means the pipe is shown and reset the count.
        //If there are whole pipes not being shown, then do not drawn them.
        if( radiusAnimationQueue.isEmpty() )
        {
                radiusAnimationQueue.add(currentRadius.get());
        }
        currentRadius.set(radiusAnimationQueue.poll());
        if(currentRadius.get() < ConcurrentVisualizer.MIN_SIZE_FOR_RADIUS)
        {
                amountOfFacesNotDrawn++;
        }
        else
        {
                amountOfFacesNotDrawn = 0;
        }
        if(amountOfFacesNotDrawn >= amountOfFaces)
        {
                amountOfFacesNotDrawn = amountOfFaces + 1;
                return;
        }

        animate(drawable);

        //Used to cap off the end so it doesn't look empty. Only do this when
        //the pipe is drawn
        if( currentRadius.get() > ConcurrentVisualizer.MIN_SIZE_FOR_RADIUS )
        {
                gl.glPushMatrix();
                gl.glTranslatef( lastFace.get()[0][0],
                        lastFace.get()[0][1]-(currentRadius.get()),
                                lastFace.get()[0][2]);
                gl.glColor4f(initialColor[0],initialColor[1],initialColor[2], 1);
                glu.gluSphere(quadric, currentRadius.get()+1, 10, 10);
                gl.glPopMatrix();
        }

        //Access the memory and draw the pipes
        gl.glPushMatrix();
```

```
                    gl.glBindBuffer(GL2.GL_ARRAY_BUFFER, pipePointer);
                     gl.glVertexPointer(3, GL2.GL_FLOAT,
                    SIZE_OF_VERTEX_INFORMATION_IN_BYTES, START_INDEX_OF_VERTEX);
                  gl.glColorPointer(4, GL2.GL_FLOAT,
SIZE_OF_VERTEX_INFORMATION_IN_BYTES, START_INDEX_OF_COLOUR);
                    gl.glNormalPointer(GL2.GL_FLOAT,
SIZE_OF_VERTEX_INFORMATION_IN_BYTES, START_INDEX_OF_NORMAL);
                gl.glBindBuffer(GL.GL_ELEMENT_ARRAY_BUFFER, indexPointer);
                gl.glDrawElements(GL.GL_TRIANGLE_STRIP, indicies.size()  ,
GL.GL_UNSIGNED_INT, 0);
              gl.glPopMatrix();
       }

       /**
        * Cascades each face from the 0th face to the (n-1)th face for n faces.
        * This accomplishes animating the pipe. For example, the 0th face's x and
        * y coordinates are given to the 1st faces, then the 1st is given to the 2nd.
        * After the faces were cascaded, the new face is removed from the queue and
        * set to the very first face. If there is nothing in the queue, then
        * the saved last face is repeated.
        *
        * @param drawable
        */
       private void animate(GLAutoDrawable drawable)
       {
              GL2 gl = drawable.getGL().getGL2();

              //To animate the pipe, I need to get the data that is within the buffer
              //and alter it.
              gl.glBindBuffer(GL2.GL_ARRAY_BUFFER, pipePointer);
              FloatBuffer floatBuffer = gl.glMapBuffer(GL2.GL_ARRAY_BUFFER,
              GL2.GL_WRITE_ONLY).asFloatBuffer();

              //Start cascading the positions from the start (0th face) to the last
              //face.
              //When the positions are finished being cascaded down, the new face is
              //set as the 0th face. Therefore, completing the animation
              //Algorithm: Save the 0th face. (This face will be the 1st face)
              float[][] currentFace;
              float currentAlpha;
              float[][] beforeFace = getFace(floatBuffer, 0);
              float beforeAlpha = getAlphaForFace(floatBuffer, 0);

              //Algorithm: We are on the ith face, we save this face because we want
              //to set the ith face to the i-1 face. After we set ith face to the i-
              //1th
              //face, we set the old value of the ith face to the i-1th face and move
              //on.
              for( int i = 1; i < amountOfFaces; i++ )
              {
                     currentFace = getFace(floatBuffer, i);
                     currentAlpha = getAlphaForFace(floatBuffer, i);

                     if( i >= minValueAlphaFading )
```

```
                        beforeAlpha = 0.75f - (( i / (float)(amountOfFaces -
                              minValueAlphaFading) ) - ( minValueAlphaFading /
                                    (float)(amountOfFaces -
                                          minValueAlphaFading)));
                  }
                  setFace(floatBuffer, beforeFace, i);
                  setAlphaForFace(floatBuffer, beforeAlpha, i);


                  beforeFace = currentFace.clone();
                  beforeAlpha = currentAlpha;
            }

            //The Rest of the faces have been changed, now it is time
            //to alter the first face (0th face). If there is
            //nothing in the queue, then the pipe needs to do what
            //it was doing before. Hence it must play the last face.
            //Therefore something is always in the queue
            if(positionAnimationQueue.isEmpty())
            {
                  positionAnimationQueue.add(lastFace.get());
            }
            if(alphaAnimationQueue.isEmpty())
            {
                  alphaAnimationQueue.add(lastAlpha.get());
            }

            //Remove what must be played next and set it to the
            //last face. This is in case of the queue being empty
            float[][] newFace = positionAnimationQueue.poll();
            lastFace.set( newFace );
            setFace(floatBuffer,newFace,0);

            //Do the same for the alpha
            float newAlpha = alphaAnimationQueue.poll();
            lastAlpha.set( newAlpha );
            setAlphaForFace(floatBuffer, newAlpha, 0);

            gl.glUnmapBuffer(GL2.GL_ARRAY_BUFFER);
      }

      /**
       * This creates a new face for the pipe which is placed at the origin
       * of the world with a radius that is specified. After each vertex is
       * created we apply a translation to put it in the proper part of
       * the world. This returns a 2D array of primitive floats
       *
       * @param radius: The pipe's radius
       * @param x: X axis translation
       * @param y: Y axis translation
       * @param z: z axis translation
       * @return
       */
      public float[][] createNewFace(float radius, float x, float y, float z)
      {
```

```
		float[][] newFace = new
			float[AMOUNT_OF_VERTS][FLOATS_USED_PER_3D_POINT];

		for( int i = 0; i < AMOUNT_OF_VERTS; i++ )
		{
			newFace[i][0] = (float) ( radius *
				Math.sin(2*i*Math.PI/(amountOfSides))) + x;
			newFace[i][1] = (float) ( radius *
				Math.cos(2*i*Math.PI/(amountOfSides))) + y;
			newFace[i][2] = 0 + z;
		}

		return newFace;
	}

	/**
	 * This method is private because it is only supposed to be used by the
	 * object.
	 * Preferably this is called once the buffer is received and then the buffer
	 * is
	 * passed to this function. For example, when animating the pipe starts, the
	 * buffer is grabbed and then after the animation is complete, we return the
	 * buffer to memory. This method is used in between the grabbing and returning
	 * the buffer.
	 *
	 * @param buffer: A float buffer that holds all the data for this pipe.
	 * @param face: The face needed. If there are 5 sections, there are 6 faces
	 * labelled 0 to 5
	 * @return 2D array of floats [amount of verts][3 points]
	 */
	private float[][] getFace( FloatBuffer buffer, int face )
	{
		float[][] verticiesValues = new
		float[AMOUNT_OF_VERTS][FLOATS_USED_PER_3D_POINT];

		for( int i = face * AMOUNT_OF_VERTS *
			SIZE_OF_VERTEX_INFORMATION_IN_FLOATS, j =0 ;
				i < face * AMOUNT_OF_VERTS *
					SIZE_OF_VERTEX_INFORMATION_IN_FLOATS +
						(AMOUNT_OF_VERTS *
							SIZE_OF_VERTEX_INFORMATION_IN_FLOATS);
					i+=SIZE_OF_VERTEX_INFORMATION_IN_FLOATS,j++ )
		{
			verticiesValues[j][0] = buffer.get((i + 0));
			verticiesValues[j][1] = buffer.get((i + 1));
			verticiesValues[j][2] = buffer.get((i + 2));
		}

		return verticiesValues;
	}

	/**
	 * Gets the alpha values of a specific face from the buffer. Even though every
	 * vertex in the face has its own alpha value, the alpha values are the same
	 * for every vertex in a face. Therefore, I just return one value
```

```
 *
 * @param The buffer that holds the data and the face that is needed.
 */
private float getAlphaForFace( FloatBuffer buffer, int face )
{
	float alpha = -1;

	for( int i = face * AMOUNT_OF_VERTS *
		SIZE_OF_VERTEX_INFORMATION_IN_FLOATS;
			i < face * AMOUNT_OF_VERTS *
				SIZE_OF_VERTEX_INFORMATION_IN_FLOATS +
					(AMOUNT_OF_VERTS *
						SIZE_OF_VERTEX_INFORMATION_IN_FLOATS);
					i+=SIZE_OF_VERTEX_INFORMATION_IN_FLOATS)
	{
		alpha = buffer.get((i + 6));
	}

	return alpha;
}

/**
 * Sets the alpha value of a specific face to the alpha value
 * specified. The face is set to a alpha value, where the face is stored
 * in the float buffer
 * .
 * @param buffer The buffer that holds the vertex, colour and normal values
 * @param alpha The alpha value to be set
 * @param face The specific face to be altered (0 to (n-1), where n is the
 * (amount of faces )
 */
private void setAlphaForFace( FloatBuffer buffer, float alpha, int face )
{
	for( int i = face * AMOUNT_OF_VERTS *
		SIZE_OF_VERTEX_INFORMATION_IN_FLOATS;
			i < face * AMOUNT_OF_VERTS *
				SIZE_OF_VERTEX_INFORMATION_IN_FLOATS +
				(AMOUNT_OF_VERTS *
					SIZE_OF_VERTEX_INFORMATION_IN_FLOATS);
					i+=SIZE_OF_VERTEX_INFORMATION_IN_FLOATS)
	{
		buffer.put((i+6),alpha);
	}
}

/**
 * Like the getFace() method above, this method is intented for use between
 * grabbing the binded buffer and releasing the binded buffer. The reason for
 * creating the method like that is for performance. If this method bounded
 * and grabbed the buffer from memory everytime it was called, it would be
 * inefficient. This can only be called if the buffer was grabbed from memory
 * then passed as a FloatBuffer Object. Considering this is used to
 * animate the pipe, it does not change the Z coordinate.
 *
```

```
 * @param values: a 2D array of floats that hold the new face. If a face has 6
 * verts, then the array is 6 rows by 3 columns.
 * @param face: The face one wants to change. If there are 4 sections, then
 * there are 5 faces labelled 0 to 4.
 */
private void setFace( FloatBuffer buffer, float[][] values , int face )
{
	for( int i = face * AMOUNT_OF_VERTS *
		SIZE_OF_VERTEX_INFORMATION_IN_FLOATS, j =0 ;
			i < face * AMOUNT_OF_VERTS *
				SIZE_OF_VERTEX_INFORMATION_IN_FLOATS +
					(AMOUNT_OF_VERTS *
						SIZE_OF_VERTEX_INFORMATION_IN_FLOATS);
					i+=SIZE_OF_VERTEX_INFORMATION_IN_FLOATS,j++ )
	{
		buffer.put((i + 0),values[j][0]);
		buffer.put((i + 1),values[j][1]);
	}
}

/**
 * Returns true if the Pipe object has been created. This means,
 * that the graphics card holds the VBO.
 *
 * @return true or false, depending on whether the pipe has been
 * initialized in graphics card memory.
 */
public boolean isCreated()
{
	return isCreated;
}

/**
 * Returns the coordinates of the pipe where the
 * pipe was initially placed in the world. This is
 * set for the life of the object.
 *
 * @return
 */
public float[] getInitialPlacement()
{
	return initialPlacement.clone();
}

/**
 * The pipe is apart of the music visualization. The
 * visualization has pipes that split into 16 channels.
 * This method sets what channel this pipe belongs to.
 *
 * @return The channel that this pipe belongs to
 */
public int getChannel()
{
	return channel;
}
```

```
/**
 * There are 16 channels and each channel is allowed to have 3
 * pipes. This returns what pipe this pipe is.
 *
 * @return The role this pipe plays
 */
public int getPipe()
{
	return pipe;
}

/**
 * Returns the face animation queue
 *
 * @return: The face animation queue
 */
public ConcurrentLinkedQueue<float[][]> getPositionAnimationList()
{
	return positionAnimationQueue;
}

/**
 * Returns the animation queue for the alpha values
 *
 * @return: The alpha animation queue
 */
public ConcurrentLinkedQueue<Float> getAlphaAnimationList()
{
	return alphaAnimationQueue;
}

/**
 * The pipe object has animation queues that store how
 * the next first (0th) face should look at be placed. These animation
 * queues must be cleared whenever a new song is loaded. Another part of
 * the animation queues are seperate primitives for the face and alpha,
 * these last values are used whenever there is nothing inside the animation
 * queue, so we have to reset these as well.
 */
public void resetPipeAnimation()
{
	lastFace.set( createNewFace(INITIAL_RADIUS, initialPlacement[0],
		initialPlacement[1], initialPlacement[2]) );
	lastAlpha.set( 1f );
	currentRadius.set( 0f );
	positionAnimationQueue.clear();
	alphaAnimationQueue.clear();
	radiusAnimationQueue.clear();
}

/**
 * In the Pipe class, there is a sphere drawn at the very front
 * of the pipe where the new faces are placed. Therefore, to
 * draw the sphere, I have to keep track of where the first
 * face is located at. The radius is used to figure out
```

```
         * whether to draw the sphere. If the radius is too small,
         * as in it is turned off, then the sphere is not drawn.
         *
         * @param firstFaceCoordinates: The position of the first (0th) face
         * @param firstFaceRadius: The radius of the first (0th) face
         */
        public void setFirstFaceData( float firstFaceRadius )
        {
              this.radiusAnimationQueue.add(firstFaceRadius);
        }

}
```

6.7 GUI.java Source Code

```
package gui;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.Frame;
import java.awt.TextField;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.media.opengl.awt.GLCanvas;
import javax.swing.BorderFactory;
import javax.swing.DefaultListModel;
import javax.swing.DropMode;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JSlider;
import javax.swing.JTextField;
import javax.swing.JToggleButton;
import javax.swing.ListSelectionModel;
import javax.swing.SwingUtilities;


import controller.Controller;
import handlers.ListTransferHandler;
import listeners.AddMidiSongMouseListener;
import listeners.ChangeColourPaletteListener;
import listeners.ChangeVisualDisplay;
import listeners.ChangeVolumeListener;
import listeners.ListChangeListener;
import listeners.ListKeyListener;
import listeners.ListMouseListener;
import listeners.LoopingCheckBoxListener;
import listeners.MaxMSPSettingsListener;
import listeners.PlayPauseToggleButtonActionListener;
import listeners.SlideMouseListener;
import listeners.SliderTimeChangeListener;
import listeners.StopButtonListener;
import renderers.FileCellRenderer;
import visualizer.ConcurrentVisualizer;

import com.jogamp.opengl.util.FPSAnimator;

/**
 * This class initialises and displays the {@link ConcurrentVisualizer} and the MIDI
 * player GUI. It also contains thread-safe methods for altering GUI objects during
 * runtime. This consists of requesting the {@link SwingUtilities} to invoke threads
 * to update the GUI during runtime in a thread-safe way. Please see Swing's
threading
 * policy.
```

```
public GUI() throws InterruptedException
	{
	 Runnable openGL = new Runnable()
			{
				public void run()
				{
					startVisualizer();
				}
			};
	 Thread t1 = new Thread(openGL);//new Thread(openGL).start();
	 t1.start();
		t1.join();


	 //startPlayerGUI();
	 Runnable playerGUI = new Runnable()
		{
			public void run()
			{
				startPlayerGUI();
			}
		};
		Thread t2 = new Thread(playerGUI);//new Thread(openGL).start();
		t2.start();
		t2.join();
	}

	/**
	 * Creates the {@link JFrame}, which contains the MIDI player controls.
	 * <p>
	 * This is invoked by {@line #GUI()} during initialisation in a thread.
	 * Swing is not thread-safe, therefore a thred is used such that the last
	 * call is always {@link JFrame}'s {@code setVisible(boolean)}.
	 */
	private void startPlayerGUI()
	{
		//Sets up the JFrame and adds a border layout, so panels can be easily
arranged.
		frame = new JFrame();
		frame.setSize(400, 400);

	 frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	 frame.setResizable(false);
	 frame.setLayout(new BorderLayout());
	 frame.setTitle("MusicViz: Midi Player");

	 //Sets up the panels
	 buttonPanel = new JPanel();
	 sliderPanel = new JPanel();
	 playListPanel = new JPanel();
	 buttonPanel.setBorder(BorderFactory.createTitledBorder("Controls"));
	 sliderPanel.setBorder(BorderFactory.createTitledBorder("Song Navigation"));
	 playListPanel.setBorder(BorderFactory.createTitledBorder("Playlist"));

	 //adds the panels to the frame
```

```
frame.add(buttonPanel,BorderLayout.NORTH);
frame.add(sliderPanel,BorderLayout.CENTER);
frame.add(playListPanel, BorderLayout.SOUTH);

//Sets up Buttons
playPauseButton = new JToggleButton("Play ");
stopButton = new JButton("Stop");
loopCheckBox = new JCheckBox("Loop");

totalTime = "0:00";

//Sets a text area for the minutes and seconds in the song
//we do not want the user to modify this either, so set it to false.
songPositionNumber = new TextField("0:00/"+totalTime,8);
songPositionNumber.setEditable(false);

//Set up the index slider and set start point to zero
//Also the slider is not enabled. Because there is no song added
slider = new JSlider();
slider.setValue(0);
slider.setSnapToTicks(true);
slider.setEnabled(false);
slider.setMinimum(0);


//Sets up the playlist
listModel = new DefaultListModel();
playList = new JList(listModel);
playList.setSelectionMode(ListSelectionModel.SINGLE_INTERVAL_SELECTION);
playList.setDragEnabled(true);
playList.setDropMode(DropMode.INSERT);

scrollPane = new JScrollPane(playList);
scrollPane.setPreferredSize(new Dimension(380,190));

//For the menu
menuBar = new JMenuBar();
fileMenu = new JMenu("File");
optionsMenu = new JMenu("Options");
openMidiItem = new JMenuItem("Load Midi");
changeColourMenuItem = new JMenuItem("Change Colours");
changeDisplayMenuItem = new JMenuItem("Change Display");
changeMaxMSPCommunication = new JMenuItem("MaxMSP Communication Settings");
changeVolumes = new JMenuItem("Change Volume");

//Adds the menus to menus and to the frame
fileMenu.add(openMidiItem);
optionsMenu.add(changeColourMenuItem);
optionsMenu.add(changeDisplayMenuItem);
optionsMenu.add(changeMaxMSPCommunication);
optionsMenu.add(changeVolumes);
menuBar.add(fileMenu);
menuBar.add(optionsMenu);
frame.setJMenuBar(menuBar);
```

```
        //Adds the buttons and the slider to appropriate panels.
        //The buttons are added to the north panel and the slider to the south
        buttonPanel.add(playPauseButton);
        buttonPanel.add(stopButton);
        buttonPanel.add(loopCheckBox);
        sliderPanel.add(songPositionNumber);
        sliderPanel.add(slider);
        playListPanel.add(scrollPane);

        frame.setEnabled(false);

        System.out.println("Variables for Player Initialized Correctly");
        frame.setVisible(true);
    }

    /**
     * This method adds listeners for the GUI.
     * <p>
     * The listeners are essential for the functionality because they
     * communicate with the front and back-ends of the program, therefore
     * they are given the {@link Controller} to use.
     * <p>
     * @param controller
     */
    public void addListeners(Controller controller)
    {
        stopButton.addActionListener( new StopButtonListener(controller) );

        playList.addMouseListener(new ListMouseListener(controller));
        playList.addKeyListener( new ListKeyListener(controller) );
        playList.setTransferHandler(new ListTransferHandler(controller));
        playList.addListSelectionListener( new ListChangeListener(controller) );
        playList.setCellRenderer( new FileCellRenderer(controller) );

        //saveMenuItem.addMouseListener(new SavePlaylistMouseListener(controller));
        //loadMenuItem.addMouseListener( new LoadPlayListMouseListener(controller));

        openMidiItem.addMouseListener( new AddMidiSongMouseListener(controller) );
        playPauseButton.addActionListener( new
PlayPauseToggleButtonActionListener(controller) );
        loopCheckBox.addActionListener( new LoopingCheckBoxListener() );
        changeColourMenuItem.addMouseListener(new
ChangeColourPaletteListener(controller));
        changeDisplayMenuItem.addMouseListener( new ChangeVisualDisplay(controller)
);
        changeMaxMSPCommunication.addMouseListener( new
MaxMSPSettingsListener(controller));
        changeVolumes.addMouseListener( new ChangeVolumeListener(controller) );

            SlideMouseListener sml = new SlideMouseListener(controller);
            slider.addMouseListener( sml );
            slider.addMouseMotionListener( sml );
            slider.addChangeListener( new SliderTimeChangeListener(controller) );
            System.out.println("Player GUI: Initialized Correctly");
            frame.setEnabled(true);
```

```
}

        /**
         * Creates the OpenGL visualiser.
         * <p>
         * This is invoked by {@line #GUI()} during initialisation in a thread.
         * Swing is not thread-safe, therefore a thred is used such that the last
         * call is always {@link JFrame}'s {@code setVisible(boolean)}.
         */
        private void startVisualizer()
        {
          openGLFrame = new Frame("Visualizer");
          canvas = new GLCanvas();
          visualizer = new ConcurrentVisualizer();
             canvas.addGLEventListener(visualizer);
             canvas.addMouseMotionListener(visualizer);
             canvas.addMouseWheelListener(visualizer);
             canvas.addKeyListener(visualizer);
             openGLFrame.add(canvas);
             openGLFrame.setSize(1024, 768);
             animator = new FPSAnimator(canvas,60);
             openGLFrame.addWindowListener(
                       new WindowAdapter()
                       {
                              public void windowClosing(WindowEvent e)
                              {
                                     new Thread(
                                                   new Runnable()
                                                   {
                                                          public void run()
                                                          {
                                                                 System.exit(0);
                                                          }
                                                   }).start();
                              }
                       });
             visualizer.setDaemon(true);
             ((Thread) visualizer).start();
             animator.start();
             System.out.println("Concurrent Visualizer Initialized Correctly");
             openGLFrame.setVisible(true);
        }

        /**
         * Updates the maximum value for the {@code JSlider}.
         * <p>
         * This is done through a call to the {@link SwingUtilities},
         * which queues an update thread. This is needed for updates to
         * {@link JFrame} to be thread-safe.
         *
         * @param maxTimeInSeconds the maximum value the slider has
         */
        public void setMaximumValueForSlider( final int maxTimeInSeconds )
        {
```

```
Runnable setMaximum =
	new Runnable()
		{
		public void run()
		{
			slider.setMaximum(maxTimeInSeconds);
		}
		};
	SwingUtilities.invokeLater(setMaximum);
}

/**
 * Updates the current value for the {@code JSlider}.
 * <p>
 * This is done through a call to the {@link SwingUtilities},
 * which queues an update thread. This is needed for updates to
 * {@link JFrame} to be thread-safe.
 *
 * @param maxTimeInSeconds the maximum value the slider has
 */
public void setCurrentValueForSlider( final int currentValue )
{
	Runnable setCurrentValue =
	new Runnable()
		{
		public void run()
		{
			slider.setValue(currentValue);
		}
		};
	SwingUtilities.invokeLater(setCurrentValue);
}

/**
 * Enables the {@code JSlider}, such that it can be used.
 * <p>
 * This is done through a call to the {@link SwingUtilities},
 * which queues an update thread. This is needed for updates to
 * {@link JFrame} to be thread-safe.
 * <p>
 * @param maxTimeInSeconds the maximum value the slider has
 */
public void enableSlider()
{
	Runnable enableSlider =
	new Runnable()
		{
		public void run()
		{
			slider.setEnabled(true);
		}
		};
	SwingUtilities.invokeLater(enableSlider);
}
```

```
/**
 * Disables the {@code JSlider}, such that it cannot be used.
 * <p>
 * This is done through a call to the {@link SwingUtilities},
 * which queues an update thread. This is needed for updates to
 * {@link JFrame} to be thread-safe.
 * <p>
 * @param maxTimeInSeconds the maximum value the slider has
 */
public void disableSlider()
{
	Runnable disableSlider =
	new Runnable()
		{
		public void run()
		{
			slider.setEnabled(false);
		}
		};
	SwingUtilities.invokeLater(disableSlider);
}

/**
 * Updates the {@link JTextField} to display the current time value.
 * <p>
 * This is done through a call to the {@link SwingUtilities},
 * which queues an update thread. This is needed for updates to
 * {@link JFrame} to be thread-safe.
 * <p>
 * @param time the current time
 */
public void updateTimer(final String time)
{
	Runnable updateTimer =
		new Runnable()
			{
			public void run()
			{
				songPositionNumber.setText(time+"/"+totalTime);
			}
			};
	SwingUtilities.invokeLater(updateTimer);
}

/**
 * Enables/Disables {@link #frame}.
 * <p>
 * This is done through a call to the {@link SwingUtilities},
 * which queues an update thread. This is needed for updates to
 * {@link JFrame} to be thread-safe.
 * <p>
 * @param enabled
 */
public void setEnabledPlayerFrame(final boolean enabled )
{
```

```
            Runnable setEnabled =
            new Runnable()
                  {
                  public void run()
                  {
                        frame.setEnabled(enabled);
                  }
                  };
            SwingUtilities.invokeLater(setEnabled);
      }

      /**
       * Returns the {@link #visualizer} object.
       * <p>
       * @return the {@link #visualizer} object
       */
      public ConcurrentVisualizer getVisualizer()
      {
            return visualizer;
      }

      /**
       * Pauses the {@link #visualizer}'s {@link #animator}.
       * <p>
       * The {@link #animator} is the main object that times the {@link
#visualizer}.
       * <p>
       * This is done through a call to the {@link SwingUtilities},
       * which queues an update thread. This is needed for updates to
       * {@link JFrame} to be thread-safe.
       */
      public void pauseAnimator()
      {
            Runnable pauseAnimator =
            new Runnable()
                  {
                  public void run()
                  {
                        animator.pause();
                  }
                  };
            SwingUtilities.invokeLater(pauseAnimator);
      }

      /**
       * Resumes the {@link #visualizer}'s {@link #animator}.
       * <p>
       * The {@link #animator} is the main object that times the {@link
#visualizer}.
       * <p>
       * This is done through a call to the {@link SwingUtilities},
       * which queues an update thread. This is needed for updates to
       * {@link JFrame} to be thread-safe.
       */
      public void resumeAnimator()
```

```
	{
		Runnable resumeAnimator =
		new Runnable()
			{
			public void run()
			{
				animator.resume();
			}
			};
		SwingUtilities.invokeLater(resumeAnimator);
	}

	/**
	 * Returns the {@link #loopCheckBox} object, which can be altered.
	 * <p>
	 * This is not a thread-safe call, when changing the object. However,
	 * this object is only read, therefore it is okay.
	 * <p>
	 * @return the checkbox, which states if the playlist should repeat.
	 */
	public JCheckBox getLoopCheckBox()
	{
		return loopCheckBox;
	}

	/**
	 * Selects or deselect the {@link GUI#playPauseButton}.
	 * <p>
	 * This is done through a call to the {@link SwingUtilities},
	 * which queues an update thread. This is needed for updates to
	 * {@link JFrame} to be thread-safe.
	 * <p>
	 * @param isSelected true to select the {@link #playPauseButton}, false to
deselect
	 */
	public void setSelectedJToggleButton(final boolean isSelected )
	{
		Runnable setSelected =
		new Runnable()
			{
			public void run()
			{
				playPauseButton.setSelected(isSelected);
			}
			};
		SwingUtilities.invokeLater(setSelected);
	}

	/**
	 * Sets the text for the {@link GUI#playPauseButton}.
	 * <p>
	 * This is done through a call to the {@link SwingUtilities},
	 * which queues an update thread. This is needed for updates to
	 * {@link JFrame} to be thread-safe.
	 * <p>
```

```
	 * @param text the text to update the {@link GUI#playPauseButton}
	 */
	public void setTextJToggleButton(final String text )
	{
		Runnable setText =
		new Runnable()
			{
			public void run()
			{
				playPauseButton.setText(text);
			}
			};
		SwingUtilities.invokeLater(setText);
	}

	/**
	 * Returns the object, which represents the playlist.
	 * <p>
	 * @return {@link #playList}
	 */
	public JList getPlaylist()
	{
		return playList;
	}

	/**
	 * Updates the total time displayed. This does not update the {@link #frame}.
	 * To update the {@link #frame}, call {@link #updateTimer(String)}.
	 * <p>
	 * @param time the total time
	 */
	public void updateTotalTime( String time )
	{
		totalTime = time;
	}
}
```

6.8 Pre-study Questionnaire

Purpose of pre-study questionnaire: The purpose of this question is to collect general information about you and your music preferences. It should take less than 5 minutes to complete this questionnaire.

1. **Please indicate your age: (please check one)**

   ❑ 18 – 24

- ❑ 25 – 34
- ❑ 35 – 44
- ❑ 45 – 54
- ❑ 55 – 64
- ❑ 65 +

2. Please indicate your gender: (please check one)

- ❑ Male
- ❑ Female

3. What is your highest level of education completed? (Please check one)

- ❑ No formal education
- ❑ Elementary school
- ❑ High School
- ❑ College
- ❑ University
- ❑ Graduate School

4. How often do you use a computer? (Please check one)

- ❑ Everyday
- ❑ Every 2 – 3 days
- ❑ Once a week
- ❑ Once a month
- ❑ Never

5. How often do you listen to music? (Please check one)

- ❑ Everyday
- ❑ Every 2 – 3 days
- ❑ Once a week
- ❑ Once a month

- ❑ Never

6. What types of music do you listen to? (Please check all that apply)

- ❑ Rock
- ❑ Rap/Hip-hop
- ❑ Country
- ❑ Classical
- ❑ Metal
- ❑ Other:_____

7. How do you experience music? (Please check all that apply, only check the ones in the list)

- ❑ iPod/MP3 Player/Portable CD player
- ❑ Computer speaker system
- ❑ Home theatre system
- ❑ Stereo
- ❑ Live concerts
- ❑ I don't listen to music

8. Why do you listen to music? (Please check most important one)

- ❑ Enjoyment
- ❑ Relaxation
- ❑ Therapy
- ❑ Emotional Experience
- ❑ Distraction
- ❑ I don't listen to music
- ❑ Other:_____

9. What do you do most often when listening to music? (Please check one):

- ❑ House Work (cleaning/cooking/laundry)
- ❑ Driving

- ❑ Work
- ❑ Exercise (running/walking/etc)
- ❑ Have a meal
- ❑ I don't do anything when listening to music
- ❑ Other:_____

10. How often do you use closed captioning or subtitles

- ❑ Always
- ❑ Sometimes
- ❑ Never

11. How do you identify yourself? (Please check one only):

- ❑ Deaf
- ❑ Deafened
- ❑ Hard of Hearing
- ❑ Hearing
- ❑ Cochlear Implant

6.9 After-each song Questionnaire

Purpose of the intermediate questionnaire: The purpose of this questionnaire is to collect your feedback about your experience of the song that you just viewed. This feedback consists of the enjoyability, the energy and the emotions that the song portrayed to you as well as your understanding of the various visual elements. It should take about 5 minutes to complete these questions.

1. Was the experience of VITA system enjoyable (circle one)?

Not enjoyable at all Not that enjoyable Neutral Somewhat enjoyable Enjoyable

Happy 

Weak						**Strong**	
1	2	3	4	5	6	7	N/A

Sad 

Weak						**Strong**	
1	2	3	4	5	6	7	N/A

Angry 

Weak						**Strong**	
1	2	3	4	5	6	7	N/A

Fear 

Weak						**Strong**	
1	2	3	4	5	6	7	N/A

1. How distracted were you from the visualization? (Please check one)

- ❑ I was very distracted from the visualization
- ❑ I was a little distracted from the visualization
- ❑ I did not notice
- ❑ I was somewhat focused on the visualization
- ❑ I was not distracted from the visualization

Why?

2. **How distracted were you from the Emoti-Chair? (Please check one)**
   - ❑ I was very distracted from the Emoti-Chair
   - ❑ I was a little distracted from the Emoti-Chair
   - ❑ I did not notice
   - ❑ I was somewhat focused on the Emoti-Chair
   - ❑ I was not distracted from the Emoti-Chair

   Why?

3. **What was your level of focus on the whole system? (Please check one)**
   - ❑ I focused only on visualization and did not focus on the Emoti-Chair
   - ❑ I focused mostly on visualization and focused a little on the Emoti-Chair
   - ❑ I focused equally on the visualization and the Emoti-Chair
   - ❑ I focused mostly on the Emoti-Chair and focused a little on the visualization
   - ❑ I focused only on the Emoti-Chair and did not focus on the visualization

4. **Please check your level of agreement/disagreement with the following statements:**

	Agree	Somewhat agree	Don't	Somewhat	Disagree

			care	disagree	
The movement on the screen was hard to watch.					
The patterns and shapes on screen were pleasant to watch.					
The vibrations helped me understand the visualization.					
The visualization helped me understand the vibrations.					
The vibrations were too strong.					
The vibration up and down my back felt pleasant.					
The pattern of vibrations on my back was confusing.					

6.10 Post-study Questionnaire

VITA Post-Study Questionnaire

Purpose of the post-study questionnaire: The purpose of this questionnaire is to understand what each visual piece of the visualization means to you, the difficulty of comprehending the visualization as well as your likes and dislikes of the experience.

1. **Rate your level of enjoyment using the VITA? (Please select one):**

Not enjoyable Not that enjoyable Neutral Somewhat enjoyable Enjoyable

2. What does the brightness in the visualization mean to you? (Please select one):

- ❑ Intensity
- ❑ Frequency
- ❑ Beat
- ❑ Grouping
- ❑ Other:_____

3. What do the shapes in the visualization mean to you? (Please select one):

- ❑ Intensity
- ❑ Frequency
- ❑ Beat
- ❑ Grouping
- ❑ Other:_____

4. What do the different heights of the pipes on the screen mean to you? (Please select one):

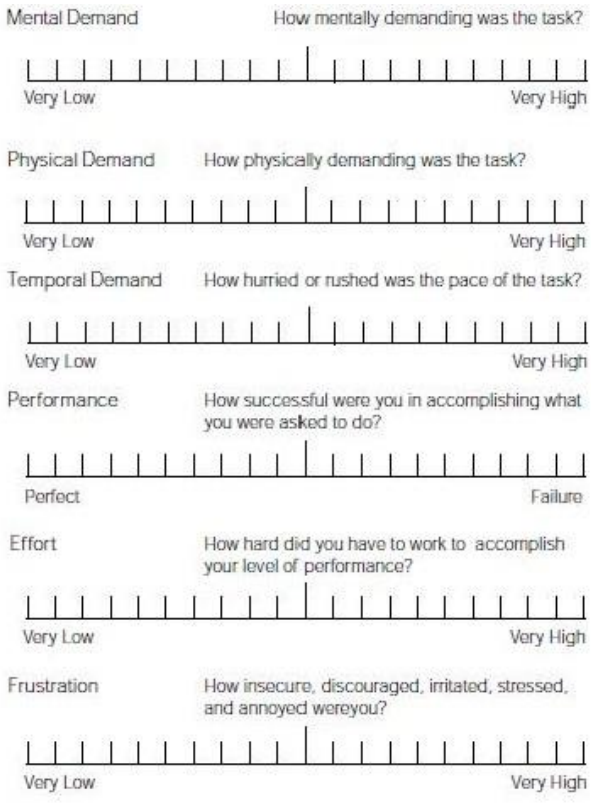
- ❑ Intensity
- ❑ Frequency
- ❑ Beat
- ❑ Group
- ❑ Other:_____

5. Overall, please rate your level of agreement for each of the following:

	Agree	Somewhat Agree	Don't Care	Somewhat Agree	Disagree
The vibrations helped me					

understand the visualization					
The visualization helped me understand the vibrations					
I was distracted throughout the experience					
The experience was comfortable					

6. Please indicate your experience in determining what was happening with the VITA system:



7. What did the circles mean to you?

8. **What did the pipes mean to you?**
9. **What did the movement mean to you?**
10. **What do the colours mean to you?**
11. **What did the different vibrations mean to you?**
12. **What did the movement up and down your back mean?**
13. **What did the different strengths of vibrations mean to you?**
14. **What did you like most about VITA?**
15. **What did you like least about VITA?**

6.11 Letter of Ethics Approval

RYERSON UNIVERSITY
RESEARCH ETHICS BOARD

To: Michael Pouris
Computer Science
Re: REB 2012-205: VITA: Visually Immersion and Tactile Animation
Date: July 19, 2012

Dear Michael Pouris,

The review of your protocol REB File REB 2012-205 is now complete. The project has been approved for a one year period. Please note that before proceeding with your project, compliance with other required University approvals/certifications, institutional requirements, or governmental authorizations may be required.

This approval may be extended after one year upon request. Please be advised that if the project is not renewed, approval will expire and no more research involving humans may take place. If this is a funded project, access to research funds may also be affected.

Please note that REB approval policies require that you adhere strictly to the protocol as last reviewed by the REB and that any modifications must be approved by the Board before they can be implemented. Adverse or unexpected events must be reported to the REB as soon as possible with an indication from the Principal Investigator as to how, in the view of the Principal Investigator, these events affect the continuation of the protocol.

Finally, if research subjects are in the care of a health facility, at a school, or other institution or community organization, it is the responsibility of the Principal Investigator to ensure that the ethical guidelines and approvals of those facilities or institutions are obtained and filed with the REB prior to the initiation of any research.

Please quote your REB file number (REB 2012-205) on future correspondence.

Congratulations and best of luck in conducting your research.

Nancy Walton, Ph.D.
Chair, Research Ethics Board

6.12 General Midi Specification Description

6.12.1 Description and assembly of Pitch-Bend Messages

Pitch-bend messages target all notes played in a channel and alters them by either increasing or decreasing the pitches. This function is synonymous with a guitar's whammy bar. Pitch-bend messages, as shown in Table 3 (in appendix), have a status between 224 and 239 and contain two extra bytes. They are similar to note-on/off messages; however, they differ in three ways: (1) pitch-bend messages target all notes being played in an entire channel; (2) the second and third bytes are related; and (3) the MIDI sequence must pre-set system parameters before utilizing pitch-bends. The two parameters are: (1) the initial-pitch settings in the channel; and (2) the sensitivity of the pitch-bend. The sensitivity is specified as musical semi-tones. Each song independently and automatically sets these parameters. Pitch-bend settings are numbers, which range between 0 (0x0000) and 16383 (0x3FFF). A song's initial pitch setting is "no pitch-bend", which is a value of 8192 [85]. A pitch value of 0 means "bend the note as low as possible" [85] and a pitch bend of 16383 means "bend the note as high as possible" [85].

Once the initial parameters are set, the MIDI system can use pitch-bend messages. Based on Table 3, a pitch-bend message is three bytes. The first byte is the status byte and the second and third bytes indicate the pitch-bend. The third byte signifies the most significant byte (MSB) and the second byte specifies the least significant byte (LSB). To combine the MSB and LSB into a single number, the MSB is left-shifted by seven and logically XORed with the LSB. This results with a new pitch setting for the entire channel. This is shown in Figure 71.

```
int pitchBend = (m[2] & 0xff) << 7 | (m[1] & 0xff);
```

Figure 71: Assembling a pitch-bend message from two bytes

The pitch-bend value is the new pitch in the channel, which is relative to the initial settings. As previously mentioned, the initial pitch setting is 0x2000. The difference between the initial value and the new value is divided by the maximum value (0x3FFF). This result is the percentage by which the pitches are bent.

Finally this percentage is multiplied by the sensitivity of the pitch-bend. The algorithm is shown in Equation 6.

Equation 6: Calculating a new pitch value

$$\frac{pitchBend - 0x2000}{0x3FFF} * sensitivity$$

As an example, a pitch-bend of 0x3000 is received with the initial value of 0xC for the sensitivity. This computes to an overall pitch-bend of 3.00018 semi-tones for each note in the channel.

In addition, MIDI information is specified as unsigned bytes; however, Java does not have unsigned types. A signed number is logically ANDed with 0xFF to convert from a signed to unsigned byte. MusicViz is designed to parse the incoming MIDI messages just described and translate the data, which represents auditory music, into a visual representation.

6.12.2 Java MIDI Playback Structure

Table 25: MIDI classes and interfaces in the Java 6 API [86]

Classes	Interfaces
Instrument	ControllerEventListener
MetaMessage	MetaEventListener
MidiDevice.Info	MidiChannel
MidiEvent	*MidiDevice
MidiFileFormat	*Receiver
*MidiMessage	*Sequencer
*MidiSystem	Soundbank
Patch	*Synthesizer
*Sequence	*Transmitter

Sequencer.SyncMode	
ShortMessage	
SoundBankResource	
SysexMessage	
Track	
VoiceStatus	

The *Receiver* interface allows MusicViz to receive every *MIDIMessage* in real-time through the *send* method regardless of the MIDI *Transmitter*. MusicViz utilizes multiple *Receiver* objects to catch and process the MIDI messages for OpenGL visualization.

The classes and interfaces directly used in the visual translation process are the *MidiSystem*, *MidiMessage*, *Receiver*, *Sequence*, *Sequencer*, *Transmitter* and *Synthesizer*. provides The *MidiSystem* class provides access to the operating system's installed MIDI input and output ports. *MidiSystem* is used to retrieve the sound *Synthesizer* and *Sequencer* from the operating system. The *Sequencer* is either a hardware or software device on the system that plays a MIDI *Sequence*. A *Synthesizer* is connected to the output of a MIDI *Sequencer*, where the *Synthesizer* generates MIDI sounds based on the *Soundbank*. The *Sequencer* and *Synthesizer* classes implement the *MidiDevice* interface, which contain input and output ports. Input ports are described as receivers (*Receiver* interface) and output ports are transmitters (*Transmitter* interface). Each *MidiDevice* can have multiple input and output ports, to which other hardware and software MIDI devices connect. Figure 72 is an example from Java, which describes creating the MIDI system and loading a MIDI song.

```
//Acquire playback sequencer from the system
Sequencer sequencer = MidiSystem.getSequencer(false);
sequencer.open();

//Aqcuire sound generator from the system
Synthesizer synthesizer = MidiSystem.getSynthesizer();
synthesizer.open();

/*
 * As the sequencer plays the MIDI sequence, the
 * synthesizer must generate the sound. Therefore
 * the sequencer's output must be the synthesizer's
 * input.
 */

//Acquire sequencer's output port and synthesizer's
//input port
Transmitter seqOutPort = sequencer.getTransmitter();
Receiver synthInPort = synthesizer.getReceiver();

//Connect the ports
seqOutPort.setReceiver(synthInPort);

//Acquire sequence from a MIDI file
Sequence sequence = MidiSystem.getSequence(new File("Metallica-Fade To
Black.mid"));

//Load sequence into sequencer for playback and play song
sequencer.setSequence(sequence);
sequencer.start();
```

Figure 72: Creating a Java MIDI system for audio playback

The *MidiDevice* interface allows for any external MIDI hardware and/or software to connect to the MIDI *Synthesizer* and *Sequencer*. This is accomplished through *Transmitter* and *Receiver* interfaces. For example, a class implementing the *Receiver* interface can receive all MIDI messages played in software (see Figure 73).

```
Transmitter customTransmitter = sequencer.getTransmitter();
customTransmitter.setReceiver(new Receiver() {

	public void send(MidiMessage message, long time)
	{
		System.out.println("Message received");
	}

	public void close()
	{
	}
});
```

Figure 73: Attaching a MIDI *Receiver* to a MIDI *Transmitter*

6.13 MusicViz's Program Initialization and OpenGL Structure

Each face contains six vertices arranged as a pentagon. Each face has a position in 3D space. Its x, y and z-coordinates are calculated as three floats, its colour (RGBA) is calculated as four floats and its normal is calculated as three floats. The normal and colour information are utilized for OpenGL's lighting equation. Each face occupies 40 bytes (3 floats + 4 floats + 3 floats = 10 floats) of memory.

To allocate a VBO, the amount of memory to utilize must be known in advance for each pipe. During initialization, fifteen sets of two pipes are created with predefined parameters such as the length of the pipe. This includes the amount of faces and the distance between each face. The memory consumption is based on the number of faces created and is specified in bytes (see Figure 74, where *AMOUNT_OF_VERTS* is four, *FLOATS_USED_PER_3D_POINT* is three and *FLOATS_USED_PER_COLOUR* is four).

```
int byteSizeForVerticies = amountOfFaces * AMOUNT_OF_VERTS * FLOATS_USED_PER_3D_POINT
     * (Float.SIZE/Byte.SIZE);

int byteSizeForColors = amountOfFaces * AMOUNT_OF_VERTS * FLOATS_USED_PER_COLOUR *
     (Float.SIZE/Byte.SIZE);

int byteSizeForNormals = amountOfFaces * AMOUNT_OF_VERTS * FLOATS_USED_PER_3D_POINT *
     (Float.SIZE/Byte.SIZE);
```

Figure 74: Calculating memory footprint for a single pipe

A pipe containing 300 faces, with 6 vertices per face and 3 floats per vertex utilizes 21600 bytes (300*6*3=5400 floats * 4 bytes/float) for vertex storage, 28800 bytes (300*6*4=7200 floats *4 bytes/float) for colour storage and another 21600 bytes for normal storage. Each pipe uses 72000 bytes of information, which is 70.3125 Kilobytes of space. Considering there are 30 pipes, 2.06 MB of memory is utilized. Each pipe is loaded into its own separate VBO and initialized separately.

Once the pipe's memory footprint is calculated, the buffer is created by first retrieving a pointer to a specific memory location. The pointer is bound for use and the amount of memory is allocated at the location. The memory is initially accessed for write operations through a byte pointer in C; however, Java allows casting to high-level data-types such as a float-buffer. This is shown in Figure 75.

```
gl.glGenBuffers(1, pointers, 0);

gl.glBindBuffer(GL2.GL_ARRAY_BUFFER, pointers[0]);

gl.glBufferData(GL2.GL_ARRAY_BUFFER, byteSizeForVerticies + byteSizeForColors +
     byteSizeForNormals, null, GL2.GL_DYNAMIC_DRAW);

FloatBuffer floatBuffer = gl.glMapBuffer(GL2.GL_ARRAY_BUFFER,
     GL2.GL_WRITE_ONLY).asFloatBuffer();
```

Figure 75: Allocating memory for a pipe using Figure 74

The code to create each face and load the data into memory is long, yet simple to follow. The code is shown in the *createPipe* method in *ConcurrentPipe.java*. The memory layout of storing a single vertex, its colour and normal data is shown in Table 26.

Table 26: Data layout of a single vertex in graphics memory

Data Layout for a single Vertex (40 bytes long, 0 to 39)									
Vertex (X,Y,Z) (0^{th} byte to 11th)			Colour (RGBA) (12^{th} byte to 27^{th})				Normal (X, Y, Z) (28^{th} byte to 39th)		
X	Y	Z	R	G	B	A	X	Y	Z

The memory structure of a pipe composed of *n* vertices is shown in Table 27, where vertex, colour and normal are specified by *V*, *C* and *N* respectively. Each vertex and the associated data are linearly stored in memory.

Table 27: Memory structure of a pipe composed of *N* vertices in graphics memory

Pipe composed of *N* vertices												
Vertex 1			Vertex 2			Vertex 3				Vertex *N*		
V_1	C_1	N_1	V_2	C_2	N_2	V_3	C_3	N_3	…	V_n	C_n	N_n

Loading the data into memory does not render the pipe. The VBO must contain instructions how to render the linear data. Figure 76 shows a section of a pipe with two faces, where each face contains 6 vertices. The first vertex is shown in green. The vertices are labeled as they are stored sequentially in memory.

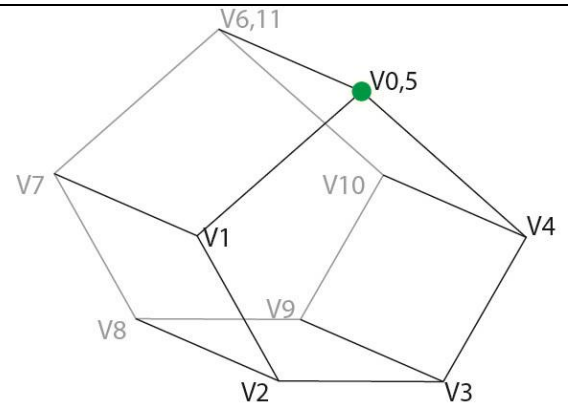


Figure 76: Vertices as stored in memory for a two-sided pipe

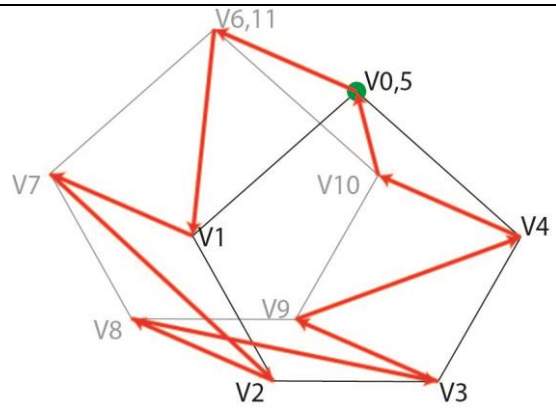


Figure 77: Visiting order of vertices to render pipe

Storing vertices in a linear fashion does not provide enough information for rendering. The simplest method of drawing this geometric structure is through the use of triangle-strips. This is specified to the VBO in a separate list dictating the sequence of vertices to visit. The path used to draw the example in Figure 76 is shown in Figure 77. The starting position is also the end position shown in green. This path is stored in an index pointer.

```
gl.glBindBuffer(GL2.GL_ARRAY_BUFFER, pipePointer);
gl.glVertexPointer(3, GL2.GL_FLOAT, SIZE_OF_VERTEX_INFORMATION_IN_BYTES,
     START_INDEX_OF_VERTEX);

gl.glColorPointer(4, GL2.GL_FLOAT, SIZE_OF_VERTEX_INFORMATION_IN_BYTES,
     START_INDEX_OF_COLOUR);

gl.glNormalPointer(GL2.GL_FLOAT, SIZE_OF_VERTEX_INFORMATION_IN_BYTES,
     START_INDEX_OF_NORMAL);

gl.glBindBuffer(GL.GL_ELEMENT_ARRAY_BUFFER, indexPointer);
gl.glDrawElements(GL.GL_TRIANGLE_STRIP, indicies.size()  , GL.GL_UNSIGNED_INT, 0);
```

Figure 78: Binding and drawing the buffer

As shown in Table 26, the information for a single vertex, such as the vertex, colour and normal values are 40 bytes long and are stored sequentially. The vertex information is 3 floats and uses the first 12 bytes (0 to 11), the colour information is 4 floats and uses 16 bytes (12 to 27) and the normal information is 3 floats and uses 12 bytes (28 to 39). This information is given to the *glVertexPointer*, *glColorPointer* and *glNormalPointer* functions. Finally the index pointer containing the paths shown in Figure 77 is supplied to the *glDrawElements* method. This method draws the pipe using the data stored in memory. Figure 78 shows the procedure.

6.14 MusicViz Playback and Visualization Architecture

6.14.1 Stage 1

```
instrumentTransmitter = sequencer.getTransmitter();
beatTransmitter = sequencer.getTransmitter();
controlTransmitter = sequencer.getTransmitter();
pitchTransmitter = sequencer.getTransmitter();
synthTransmitter = sequencer.getTransmitter();

synthesizer.open();
sequencer.open();

instrumentTransmitter.setReceiver(instrumentReceiver);
beatTransmitter.setReceiver(beatReceiver);
controlTransmitter.setReceiver(controlReceiver);
pitchTransmitter.setReceiver(pitchReceiver);
synthTransmitter.setReceiver(synthesizer.getReceiver());

instrumentReceiver.setDaemon(true);
( (Thread) instrumentReceiver).start();

beatReceiver.setDaemon(true);
( (Thread) beatReceiver).start();

pitchReceiver.setDaemon(true);
( (Thread) pitchReceiver).start();

controlReceiver.setDaemon(true);
( (Thread) controlReceiver ).start();
```

Figure 79: MIDIPlayer class initialization

6.14.2 Stage 2

6.14.2.1MusicViz's Threading Model

Java's MIDI sound system performs all functions on a dedicated thread. The thread is separate from other JVM threads and is called the *Java Sound Sequencer*. All *Transmitter* and *Receiver* interfaces run on the *Java Sound Sequencer* thread.

Initially when MusicViz was developed, the *send* method processed all messages on the *Java Sound Sequencer* thread (see Figure 80). This taxed the sound system to an extent that the MIDI *Sequencer*

skipped, crashed, or stuttered. The solution consisted of offloading processing onto separate threads, which released Java's sound resources.

```
new Receiver()
{
	//Runs on the Java Sound Sequencer Thread
	public void send(MidiMessage message, long time)
	{
		//MIDI message processing
	}
	public void close(){}
}
```

Figure 80: Original Receiver object running on the Sound Sequencer Thread

The *PitchReceiver.java*, *InstrumentReceiver.java* and *BeatReceiver.java* classes now extend Java's *Thread* class. In addition, they also implement the *Receiver* interface. As a MIDI sequence plays, each message is received by the *send* method and is transferred to a separate thread for processing. In this solution, each *Receiver* acts as a MIDI *Receiver* as well as a separate thread. This solution is shown in Figure 81.

```
public class *Receiver extends Thread implements Receiver
{
	private LinkedBlockingQueue<MidiMessage> handOffQueue;

	//Runs of the Java Sound Sequencer Thread
	public void send(MidiMessage message, long timeStamp)
	{
		this.handOffQueue.put(message);
	}

	//Runs on a different thread
	public void run()
	{
		while(true)
		{
			//Poll the blocking data structure.
			//Blocks the thread from running when nothing is available
			MidiMessage message = this.handOffQueue.take();
		}
	}
}
```

Figure 81: Sending MIDI messages from the *Java Sound Sequencer* to a separate thread for processing

The Java *Sound Sequencer Thread* passes data to each processing thread using a concurrent data structure. Java's API provides numerous concurrent data-structures that are non-blocking and blocking. A *LinkedBlockingQueue* is utilized as the communication queue between the threads. It provides thread-safe operations for polling and queuing objects. The *LinkedBlockingQueue* blocks the polling thread until an object is queued. This prevents the polling thread from wasting CPU cycles, which increases performance. Each *Receiver* object in Figure 51 follows the generic outline shown in Figure 81.

6.14.2.2Beat Processing Thread

MIDI percussion presents a problem for MusicViz because it represents drums as five toroid objects, yet there are forty-seven drums in MIDI (see Table 28). The solution groups similar drums based on three criteria: (1) similarity of drum categories; (2) usage in popular western music; and (3) obscurity (see Table 8 for groupings). All MIDI bass drums are thus grouped together (see *Group 1 in Table 8*). Different snare drums are grouped together due to their similarity and usage in popular Western music (see *Group 3*). Moreover, these groupings are limited to five visual groups as too many visuals are overwhelming and hard to watch for individuals [10].

Table 28: MIDI percussion mapping [87]

Group 1		Group 5	
2nd Byte	**Percussion Type**	**2nd Byte**	**Percussion Type**
35	Acoustic Bass Drum	39	Hand Clap
36	Bass Drum 1	54	Tambourine
Group 2		56	Cowbell
41	Low Floor Tom	60	Hi Bongo
43	High Floor Tom	61	Low Bongo
45	Low Tom	62	Mute Hi Conga

47	Low-Mid Tom	63	Open Hi Conga
48	Hi-Mid Tom	64	Low Conga
49	Crash Cymbal 1	65	High Timbale
50	High Tom	66	Low Timbale
51	Ride Cymbal 1	67	High Agogo
52	Chinese Cymbal	68	Low Agogo
53	Ride Bell	69	Cabasa
55	Splash Cymbal	70	Maracas
57	Crash Cymbal	71	Short Whistle
59	Ride Cymbal 2	72	Long Whistle
Group 3		73	Short Guiro
37	Side Stick	74	Long Guiro
38	Accoustic Snare	75	Claves
40	Snare Drum 2	76	Hi Wood Block
Group 4		77	Low Wood Block
42	Closed Hi-Hat	78	Mute Cuica
44	Pedal Hi-Hat	79	Open Cuica
46	Open Hi-Hat	80	Mute Triangle
		81	Open Triangle
		58	Vibra Slap

6.14.3 Stage 3

6.14.3.1Worker Task Architecture

Before discussing the third stage's animation techniques, OpenGL's threading model must be described because it influenced MusicViz's thread architecture. OpenGL, and consequently JOGL, are single

threaded APIs, where graphics related rendering, texture loading and animation occur within a single thread. To use multiple threads within OpenGL, the thread performing OpenGL calls must be made *current*. However, simultaneous updates to the OpenGL system from multiple threads should not be allowed. Apple [78] states that OpenGL is not reentrant, such that simultaneous updates result in unpredictable behavior and crashes. Synchronizing the OpenGL context threads such that only one thread updates the shared graphics resources solves this. However, context synchronization is the most complex model for designing applications. It is only useful when multiple scenes are rendered in parallel or in graphics engines. Figure 82 shows the architecture of a program using two OpenGL contexts.

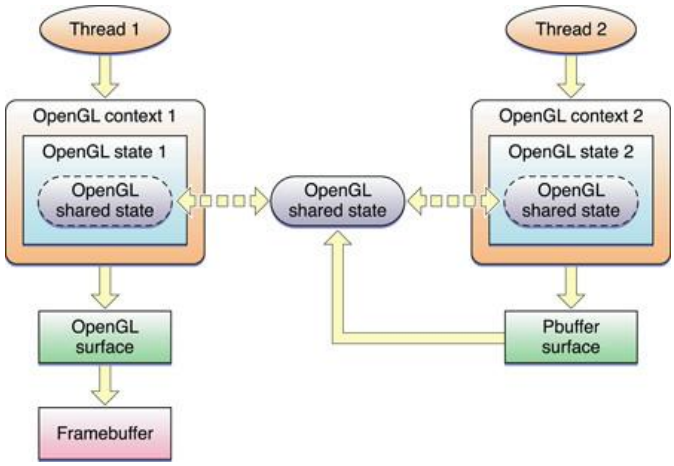


Figure 82: Two OpenGL contexts [78]

Apple [78] states that this program architecture is the most complicated and is mostly useful when rendering multiple scenes in parallel or when loading textures on-the-fly as a user walks through a virtual world [88]. This method keeps a copy of the data in memory and streams data to video memory at a proper moment to reduce lag. MusicViz does not render multiple scenes in parallel or load textures on the fly. Instead, visualization in MusicViz is decomposed into two overall processes: (1) animation processing and queuing; and (2) OpenGL rendering. Essentially these steps are non-OpenGL and OpenGL tasks, where the CPU processes one and the GPU processes the other. This architecture is referred to as *Worker Task Architecture* by Apple [78] and follows the architecture shown in Figure 83.

This architecture does not require threads to share OpenGL contexts and is simpler to implement.

MusicViz's rendering architecture implements *Worker Task Architecture*.

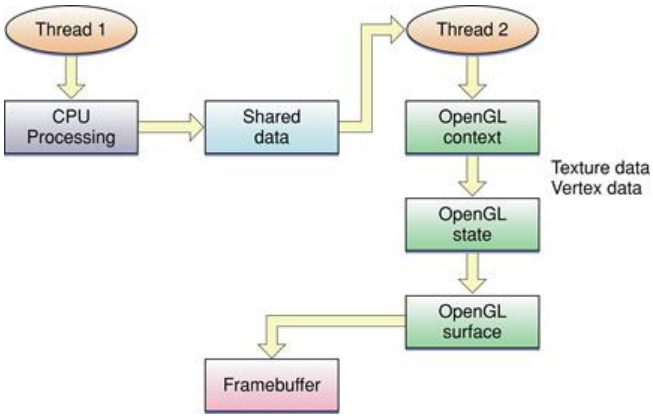


Figure 83: Worker task architecture [78]

6.15 Not Significant Per-song Results

6.15.1 Levels of Enjoyability

6.15.1.1 Within Genres

Table 29: Descriptive statistics for the levels of enjoyability for each genre (N=19)

Genre	Mean	SD
Classical	2.32	0.75
Country	2.47	0.84
Jazz	2.53	0.77
Pop	2.74	0.65
Rap	2.58	0.69
Rock	2.68	0.67

6.15.1.2Between Conditions

Table 30: Descriptive statistics for the levels of enjoyability for each condition

Condition	Mean	SD	N
1	2.37	0.89	30
2	2.71	0.46	24
3	2.58	0.73	64

6.15.1.3Within Genres and between Hearing Status Groups

Table 31: Descripvie statistics for the levels of enjoyability within genres and between hearing status groups

	Hearing Status	Mean	SD	N
Classical	Deaf	2.40	0.70	10
	Hearing	2.22	0.83	9
Country	Deaf	2.30	0.95	10
	Hearing	2.67	0.71	9
Jazz	Deaf	2.40	0.84	10
	Hearing	2.67	0.71	9
Pop	Deaf	2.50	0.85	10
	Hearing	3.00	0.00	9
Rap	Deaf	2.30	0.82	10
	Hearing	2.89	0.33	9
Rock	Deaf	2.60	0.70	10
	Hearing	2.78	0.67	9

6.15.1.4Within Genres and between Conditions

Table 32: Descriptive statistics for the levels of enjoyability within genres and between conditions

	Condition	Mean	SD	N
Classical	1	1.80	0.84	5
	2	2.50	0.58	4
	3	2.50	0.71	10
Country	1	2.60	0.89	5
	2	2.50	0.58	4
	3	2.40	0.97	10
Jazz	1	2.20	1.10	5
	2	3.00	0.00	4
	3	2.50	0.71	10
Pop	1	2.60	0.89	5
	2	2.75	0.50	4
	3	2.80	0.63	10
Rap	1	2.40	0.89	5
	2	2.75	0.50	4
	3	2.60	0.70	10
Rock	1	2.60	0.89	5
	2	2.75	0.50	4
	3	2.70	0.68	10

6.15.2 Levels of Continuous Emotions

6.15.2.1Within Genres

Table 33: Descriptive statistics for the levels of valence for each genre (N=20)

Measure	Genre	Mean	SD
Valence	Classical	6.00	1.56
	Country	6.25	2.02
	Jazz	6.00	1.49
	Pop	6.45	1.76
	Rap	6.40	1.93
	Rock	5.65	1.90

Table 34: Descriptive statistics for the levels of arousal for each genre (N=20)

Measure	Genre	Mean	SD
Arousal	Classical	5.30	1.95
	Country	5.00	2.60
	Jazz	5.55	2.37
	Pop	5.25	2.45
	Rap	5.80	2.14
	Rock	5.50	2.40

Table 35: Descriptive statistics for the levels of overbearingness for each genre (N=20)

Measure	Genre	Mean	SD
	Classical	5.00	1.62

Overbearingness	Country	4.80	2.0
	Jazz	5.45	2.06
	Pop	5.00	2.25
	Rap	5.50	1.91
	Rock	5.15	1.98

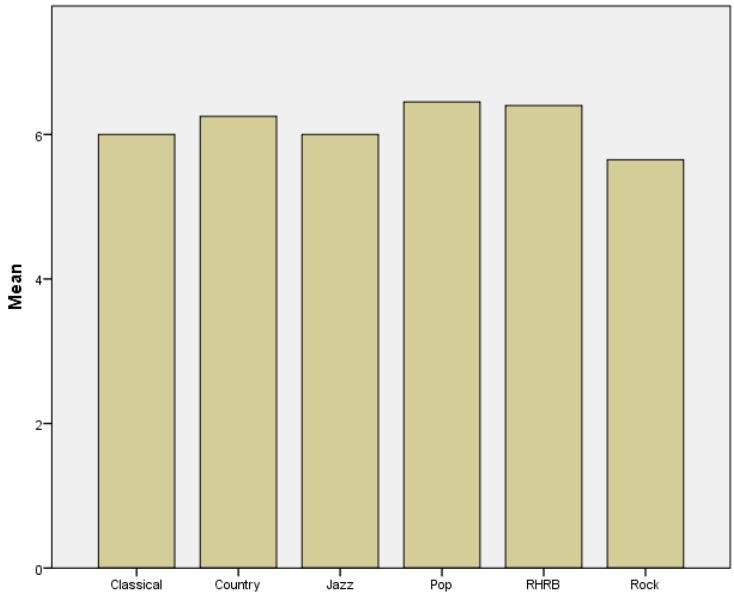


Figure 84: Descriptive statistics for the levels of valence for each genre

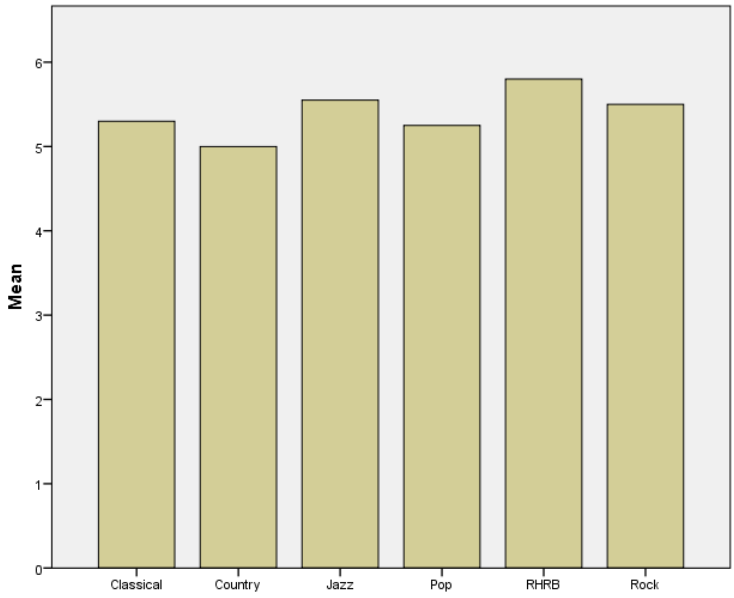


Figure 85: Descriptive statistics for the levels of arousal for each genre

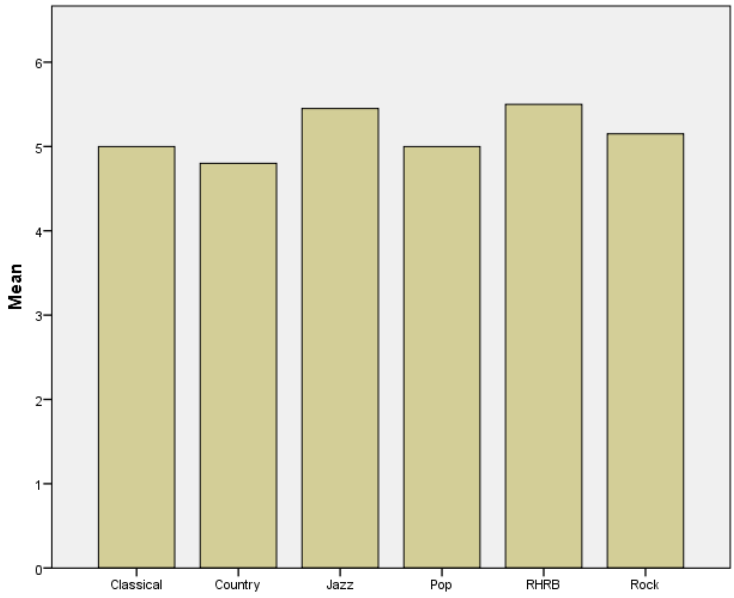


Figure 86: Descriptive statistics for the levels of overbearingness for each genre

Table 36 to Table 41 show the Spearman's Rho correlation for valence, arousal and overbearingness between each genre. Low correlations are grayed out.

Table 36: Spearman's rho correlation between the levels of valence, arousal and overbearingness for classical music (N=20, df= 18)

Correlation for Classical Music			Valence	Arousal	Overbearingness
Spearman's rho (ρ)	Valence	Correlation Coefficient	1.000	0.319	0.376
		Sig. (2-tailed)		0.171	0.102
	Arousal	Correlation Coefficient	0.319	1.000	0.533
		Sig. (2-tailed)	0.171		0.015
	Overbearingness	Correlation Coefficient	0.376	0.533	1.000
		Sig. (2-tailed)	0.102	0.015	

Table 37: Spearman's rho correlation between the levels of valence, arousal and overbearingness for country music (N=20, df=18)

Correlation for Country Music			Valence	Arousal	Overbearingness
Spearman's rho (ρ)	Valence	Correlation Coefficient	1.000	0.250	-0.161
		Sig. (2-tailed)		0.288	0.498
	Arousal	Correlation Coefficient	0.250	1.000	0.679
		Sig. (2-tailed)	0.288		0.001

	Overbearingness	Correlation Coefficient	-.161	0.679	1.000
		Sig. (2-tailed)	0.498	0.001	

Table 38: Spearman's rho correlation between the levels of valence, arousal and overbearingness for jazz music (N=20, df=18)

Correlation for Jazz Music			Valence	Arousal	Overbearingness
Spearman's rho (ρ)	Valence	Correlation Coefficient	1.000	0.447	0.300
		Sig. (2-tailed)		0.048	0.199
	Arousal	Correlation Coefficient	0.447	1.000	0.826
		Sig. (2-tailed)	0.048		0.000
	Overbearingness	Correlation Coefficient	0.300	0.826	1.000
		Sig. (2-tailed)	0.199	0.000	

Table 39: Spearman's rho correlation between the levels of valence, arousal and overbearingness for pop music (N=20, df=18)

Correlation for Pop Music			Valence	Arousal	Overbearingness
Spearman's rho (ρ)	Valence	Correlation Coefficient	1.000	0.590	0.306
		Sig. (2-tailed)		0.006	0.189

	Arousal	Correlation Coefficient	0.590	1.000	0.726
		Sig. (2-tailed)	0.006		0.000
	Overbearingness	Correlation Coefficient	0.306	0.726	1.000
		Sig. (2-tailed)	0.189	0.000	

Table 40: Spearman's rho correlation between the levels of valence, arousal and overbearingness for RHRB music (N=20, df=18)

Correlations for RHRB			**Valance**	**Arousal**	**Overbearingness**
Spearman's rho (ρ)	Valence	Correlation Coefficient	1.000	0.346	0.028
		Sig. (2-tailed)		0.135	0.905
	Arousal	Correlation Coefficient	0.346	1.000	0.363
		Sig. (2-tailed)	0.135		0.116
	Overbearingness	Correlation Coefficient	0.028	0.363	1.000
		Sig. (2-tailed)	0.905	0.116	

Table 41: Spearman's rho correlation between the levels of valence, arousal and overbearingness for rock music (N=20, df=18)

Correlations for Rock			Valence	Arousal	Overbearingness
Spearman's rho (ρ)	Valence	Correlation Coefficient	1.000	0.261	-0.146
		Sig. (2-tailed)		0.267	0.538
	Arousal	Correlation Coefficient	0.261	1.000	0.604
		Sig. (2-tailed)	0.267		0.005
	Overbearingness	Correlation Coefficient	-0.146	0.604	1.000
		Sig. (2-tailed)	0.538	0.005	

6.15.2.2Between Deaf and Hearing Groups

Table 42: Descriptive Statistics for the levels of emotions for each hearing status

Hearing Status		Valence	Arousal	Overbearingness
Deaf	Mean	6.05	5.25	5.10
	N	60	60	60
	SD	1.85	2.50	2.27
Hearing	Mean	6.20	5.55	5.20
	N	60	60	60
	SD	1.71	2.08	1.59
Total	Mean	6.13	5.40	5.15

	N	120	120	120
	SD	1.77	2.29	1.95

6.15.2.3Within Genres and between Hearing and Deaf Groups

The following tables show the descriptive statistics for the levels of valence, arousal and overbearingness between genres and hearing status groups.

Table 43: Descriptive statistics for the levels of valence for each genre and hearing status (N=20, 10 Deaf, 10 Hearing)

Measure	Genre	Hearing Status	Mean	SD
Valence	Classical	Deaf	6.20	1.55
		Hearing	5.80	1.62
	Country	Deaf	6.10	2.23
		Hearing	6.40	1.90
	Jazz	Deaf	5.90	1.97
		Hearing	6.10	0.88
	Pop	Deaf	6.10	1.91
		Hearing	6.80	1.62
	Rap	Deaf	5.80	1.87
		Hearing	7.00	1.89
	Rock	Deaf	6.20	1.93
		Hearing	5.10	1.79

Table 44: Descriptive statistics for the levels of arousal for each genre and hearing status (N=20, 10 Deaf, 10 Hearing)

Measure	Genre	Hearing Status	Mean	SD
Arousal	Classical	Deaf	5.50	1.84

		Hearing	5.10	2.13
	Country	Deaf	4.90	3.00
		Hearing	5.10	2.28
	Jazz	Deaf	5.40	2.78
		Hearing	5.70	2.00
	Pop	Deaf	5.30	2.79
		Hearing	5.20	2.20
	Rap	Deaf	5.40	2.27
		Hearing	6.20	2.04
	Rock	Deaf	5.00	2.71
		Hearing	6.00	2.06

Table 45: Descriptive statistics for the levels of overbearingness for each genre and hearing status (N=20, 10 Deaf, 10 Hearing)

Measure	Genre	Hearing Status	Mean	SD
Overbearingness	Classical	Deaf	5.40	1.65
		Hearing	4.60	1.58
	Country	Deaf	4.70	2.36
		Hearing	4.90	1.66
	Jazz	Deaf	5.30	2.45
		Hearing	5.60	1.71
	Pop	Deaf	5.00	2.75
		Hearing	5.00	1.76
	Rap	Deaf	5.70	2.11

		Hearing	5.30	1.77
	Rock	Deaf	4.50	2.51
		Hearing	5.80	1.03

6.15.2.4Within Genres and between Conditions

Table 46: Descriptive statistics for the levels of valence for each genre and condition

Measure	Genre	Condition	Mean	SD	N
Valence	Classical	1	5.20	1.10	5
		2	5.25	1.26	4
		3	6.64	1.63	11
	Country	1	5.80	1.48	5
		2	6.50	2.65	4
		3	6.36	2.16	11
	Jazz	1	5.00	1.23	5
		2	6.75	0.50	4
		3	6.18	1.66	11
	Pop	1	6.00	1.73	5
		2	6.25	1.89	4
		3	6.73	1.85	11
	Rap	1	5.40	2.19	5
		2	7.50	1.92	4
		3	6.45	1.75	11
	Rock	1	6.00	1.41	5

		2	5.25	2.50	4
		3	5.64	2.01	11

Table 47: Descriptive statistics for the levels of arousal for each genre and condition

Measure	Genre	Condition	Mean	SD	N
Arousal	Classical	1	5.00	0.71	5
		2	5.50	2.89	4
		3	5.36	2.11	11
	Country	1	5.00	2.55	5
		2	2.50	1.73	4
		3	5.91	2.43	11
	Jazz	1	5.20	2.68	5
		2	5.00	2.94	4
		3	5.91	2.21	11
	Pop	1	5.60	2.79	5
		2	3.25	1.71	4
		3	5.82	2.32	11
	Rap	1	5.60	3.44	5
		2	5.75	2.22	4
		3	5.91	1.58	11
	Rock	1	3.20	2.59	5
		2	6.25	2.50	4
		3	6.27	1.68	11

Table 48: Descriptive statistics for the levels of overbearingness for each genre and condition

Measure	Genre	Condition	Mean	SD	N
Overbearingness	Classical	1	5.20	0.84	5
		2	3.75	2.50	4
		3	5.36	1.43	11
	Country	1	4.80	2.28	5
		2	3.50	0.58	4
		3	5.27	2.10	11
	Jazz	1	5.60	2.79	5
		2	5.00	2.58	4
		3	5.55	1.70	11
	Pop	1	5.00	2.55	5
		2	3.25	1.71	4
		3	5.64	2.11	11
	Rap	1	5.00	2.55	5
		2	5.00	2.71	4
		3	5.91	1.30	11
	Rock	1	4.20	2.28	5
		2	5.25	2.06	4
		3	5.55	1.86	11

6.15.3 Levels of Discrete Emotion Ratings

6.15.3.1Within Genres

Table 49: Descriptive statistics for the discrete level of happiness for each genre (N=20)

Measure	Genre	Mean	SD
Discrete Level of Happiness	Classical	3.80	2.29
	Country	4.55	2.42
	Jazz	4.20	1.96
	Pop	4.35	2.46
	Rap	4.25	2.65
	Rock	3.75	2.51

Table 50: Descriptive statistics for the discrete level of sadness for each genre (N=20)

Measure	Genre	Mean	SD
Discrete Level of Sadness	Classical	2.05	1.93
	Country	1.75	1.92
	Jazz	2.05	1.85
	Pop	1.50	1.57
	Rap	1.65	1.73
	Rock	1.90	2.08

Table 51: Descriptive statistics for the discrete level of anger for each genre (N=20)

Measure	Genre	Mean	SD
Discrete Level	Classical	1.50	1.28

of Anger	Country	1.60	1.73
	Jazz	2.20	2.12
	Pop	1.65	1.76
	Rap	1.85	1.95
	Rock	2.15	2.37

Table 52: Descriptive statistics for the discrete level of fear for each genre (N=20)

Measure	Genre	Mean	SD
Discrete Level of Fear	Classical	1.55	1.67
	Country	1.60	1.68
	Jazz	2.15	2.23
	Pop	1.70	1.90
	Rap	1.75	1.89
	Rock	2.00	2.25

6.15.3.2Between Hearing and Deaf Groups

Table 53: Descriptive statistics for the levels of discrete emotions for each hearing status

Hearing Status		Discrete Happiness	Discrete Sadness	Discrete Anger	Discrete Fear
Deaf	Mean	3.80	1.97	1.90	1.78
	N	60	60	60	60
	SD	2.59	2.12	2.09	2.16

Hearing	Mean	4.50	1.67	1.75	1.80
	N	60	60	60	60
	SD	2.06	1.48	1.65	1.67
Total	Mean	4.15	1.82	1.83	1.79
	N	120	120	120	120
	SD	2.36	1.82	1.88	1.92

6.15.3.3Within Genres and Between Hearing and Deaf Groups

Table 54: Descriptive statistics for the levels of happiness for each genre and hearing status (N=20, 10 Deaf, 10 Hearing)

Measure	Genre	Hearing Status	Mean	SD
Discrete Level of Happiness	Classical	Deaf	3.80	2.39
		Hearing	3.80	2.30
	Country	Deaf	4.40	2.86
		Hearing	4.70	2.00
	Jazz	Deaf	3.90	2.33
		Hearing	4.50	1.58
	Pop	Deaf	3.10	2.73
		Hearing	5.60	1.35
	Rap	Deaf	3.90	2.56
		Hearing	4.60	2.84
	Rock	Deaf	3.70	3.09
		Hearing	3.80	1.93

Table 55: Descriptive statistics for the levels of sadness for each genre and hearing status (N=20, 10 Deaf, 10 Hearing)

Measure	Genre	Hearing Status	Mean	SD
Discrete Level of Sadness	Classical	Deaf	2.50	2.42
		Hearing	1.60	1.27
	Country	Deaf	1.70	1.95
		Hearing	1.80	1.99
	Jazz	Deaf	2.20	2.30
		Hearing	1.90	1.37
	Pop	Deaf	1.50	1.72
		Hearing	1.50	1.51
	Rap	Deaf	1.90	2.08
		Hearing	1.40	1.35
	Rock	Deaf	2.00	2.54
		Hearing	1.80	1.62

Table 56: Descriptive statistics for the levels of fear for each genre and hearing status (N=20, 10 Deaf, 10 Hearing)

Measure	Genre	Hearing Status	Mean	SD
Discrete Level of Fear	Classical	Deaf	1.30	1.42
		Hearing	1.80	1.93
	Country	Deaf	1.80	1.81
		Hearing	1.40	1.58
	Jazz	Deaf	2.40	2.99

		Hearing	1.90	1.198
	Pop	Deaf	1.80	2.349
		Hearing	1.60	1.43
	Rap	Deaf	1.90	2.13
		Hearing	1.60	1.71
	Rock	Deaf	1.50	2.32
		Hearing	2.50	2.17

6.15.3.4Within Genres and between Conditions

Table 57: Descriptive statistics for the discrete levels of happiness for each genre and condition

Measure	Genre	condition	Mean	SD	N
Discrete Level of Happiness	Classical	1	1.60	2.07	5
		2	3.25	2.28	4
		3	5.00	1.61	11
	Country	1	3.20	2.17	5
		2	4.75	2.87	4
		3	5.09	2.34	11
	Jazz	1	2.80	2.59	5
		2	4.25	1.26	4
		3	4.82	1.66	11
	Pop	1	3.80	2.59	5
		2	4.00	2.94	4
		3	4.73	2.41	11
	Rap	1	1.80	3.03	5

		2	5.00	3.37	4
		3	5.09	1.51	11
	Rock	1	2.80	2.78	5
		2	3.00	2.94	4
		3	4.45	2.25	11

Table 58: Descriptive statistics for the discrete levels of sadness for each genre and condition

Measure	Genre	condition	Mean	SD	N
Discrete Level of Sadness	Classical	1	1.20	1.10	5
		2	0.50	0.58	4
		3	3.00	2.05	11
	Country	1	1.20	2.17	5
		2	2.00	2.83	4
		3	1.91	1.58	11
	Jazz	1	1.20	2.17	5
		2	1.50	1.29	4
		3	2.64	1.80	11
	Pop	1	1.60	2.07	5
		2	1.00	1.41	4
		3	1.64	1.50	11
	Rap	1	0.80	1.30	5
		2	0.50	0.58	4
		3	2.45	1.81	11
	Rock	1	0.40	0.55	5

		2	1.50	1.73	4
		3	2.73	2.28	11

Table 59: Descriptive statistics for the discrete levels of anger for each genre and condition

Measure	Genre	condition	Mean	SD	N
Discrete Level of Anger	Classical	1	1.00	1.00	5
		2	1.00	1.41	4
		3	1.91	1.30	11
	Country	1	0.80	1.30	5
		2	0.50	0.58	4
		3	2.36	1.86	11
	Jazz	1	1.00	1.73	5
		2	1.75	1.71	4
		3	2.91	2.26	11
	Pop	1	0.60	0.89	5
		2	0.50	0.58	4
		3	2.55	1.86	11
	Rap	1	1.60	1.52	5
		2	1.00	0.82	4
		3	2.27	2.37	11
	Rock	1	0.60	0.89	5
		2	2.50	3.32	4
		3	2.73	2.33	11

Table 60: Descriptive statistics for the discrete levels of fear for each genre and condition

Measure	Genre	condition	Mean	SD	N
Discrete Level of Fear	Classical	1	0.60	0.89	5
		2	2.00	2.45	4
		3	1.82	1.60	11
	Country	1	1.00	1.73	5
		2	1.00	1.41	4
		3	2.09	1.70	11
	Jazz	1	2.00	2.55	5
		2	2.00	1.83	4
		3	2.27	2.41	11
	Pop	1	1.60	2.07	5
		2	0.50	0.58	4
		3	2.18	2.04	11
	Rap	1	1.60	2.07	5
		2	0.50	0.58	4
		3	2.27	2.00	11
	Rock	1	0.40	0.55	5
		2	2.50	3.00	4
		3	2.55	2.25	11

6.15.4 Levels of Agreement

6.15.4.1Within Genres

Table 61: Descriptive statistics for the levels of agreement with statement 1 for each genre (N=17)

Measure	Genre	Mean	SD
The movement on the screen was hard to watch	Classical	2.47	0.80
	Country	2.35	0.86
	Jazz	2.53	0.80
	Pop	2.65	0.79
	Rap	2.53	0.80
	Rock	2.53	0.87

Table 62: Descriptive statistics for the levels of agreement with statement 2 for each genre (N=17)

Measure	Genre	Mean	SD
The patterns and shapes on the screen were pleasant to watch	Classical	1.59	0.80
	Country	1.71	0.85
	Jazz	1.59	0.87
	Pop	1.47	0.87
	Rap	1.35	0.79
	Rock	1.59	0.87

Table 63: Descriptive statistics for the levels of agreement with statement 3 for each genre (N=17)

Measure	Genre	Mean	SD
The vibrations helped me understand the visualization	Classical	1.76	0.90
	Country	1.59	0.87
	Jazz	1.88	0.99
	Pop	1.59	0.87

	Rap	1.53	0.87
	Rock	1.88	0.99

Table 64: Descriptive statistics for the levels of agreement with statement 4 for each genre (N=17)

Measure	Genre	Mean	SD
The visualization helped me understand the vibrations	Classical	1.71	0.92
	Country	1.71	0.85
	Jazz	1.71	0.99
	Pop	1.53	0.87
	Rap	1.82	1.02
	Rock	1.88	0.99

Table 65: Descriptive statistics for the levels of agreement with statement 6 for each genre (N=17)

Measure	Genre	Mean	SD
The vibrations up and down my back felt pleasant	Classical	1.47	0.80
	Country	1.35	0.79
	Jazz	1.47	0.80
	Pop	1.35	0.70
	Rap	1.35	0.70
	Rock	1.29	0.69

Table 66: Descriptive statistics for the levels of agreement with statement 7 for each genre (N=17)

Measure	Genre	Mean	SD

The pattern of vibrations on my back was confusing	Classical	2.59	0.71
	Country	2.47	0.87
	Jazz	2.65	0.79
	Pop	2.47	0.87
	Rap	2.29	0.99
	Rock	2.53	0.87

6.15.4.2Between Hearing and Deaf Groups

Table 67: Descriptive statistics for the levels of agreement with statements 1 to 3 for each hearing status

Hearing Status		The Movement on the screen was hard to watch	The patterns and shapes on the screen were pleasant to watch	The vibrations helped me understand the visualization
Deaf	Mean	2.42	1.70	1.62
	N	60	60	58
	SD	0.85	0.93	0.90
Hearing	Mean	2.49	1.46	1.68
	N	57	57	57
	SD	0.85	0.73	0.89
Total	Mean	2.45	1.58	1.65
	N	117	117	115
	SD	0.85	0.84	0.89

Table 68: Descriptive statistics for the levels of agreement with statement 4 to 7 for each hearing status

Hearing Status		The visualization helped me understand the vibrations	The Vibrations were too strong	The vibrations up and down my back felt pleasant	The pattern of vibration on my back was confusing
Deaf	Mean	1.60	2.43	1.52	2.59
	N	58	58	58	58
	SD	0.90	0.75	0.84	0.77
Hearing	Mean	1.75	2.58	1.30	2.48
	N	57	59	60	60
	SD	0.89	0.75	0.65	0.85
Total	Mean	1.68	2.50	1.41	2.53
	N	115	117	118	118
	SD	0.89	0.75	0.75	0.81

6.15.4.3Between Conditions

Table 69: Descriptive statistics for the levels of agreement for statements 5 to 7 for each condition

Condition		The Vibrations were too strong	The vibrations up and down my back felt pleasant	The pattern of vibration on my back was confusing
1	Mean	2.54	1.57	2.50
	N	28	28	28
	SD	0.69	0.88	0.84

2	Mean	2.43	1.25	2.67
	N	23	24	24
	SD	0.79	0.61	0.70
3	Mean	2.52	1.39	2.50
	N	66	66	66
	SD	0.77	0.74	0.85
Total	Mean	2.50	1.41	2.53
	N	117	118	118
	SD	0.75	0.75	0.81

6.15.4.4Within Genres and between Hearing and Deaf Groups

Table 70: Descriptive statistics for the levels of agreement with statement 1 for each genre and hearing status

Measure	Genre	Hearing Status	Mean	SD	N
The movement on the screen was hard to watch	Classical	Deaf	2.56	0.73	9
		Hearing	2.38	0.92	8
	Country	Deaf	2.56	0.73	9
		Hearing	2.13	0.99	8
	Jazz	Deaf	2.56	0.73	9
		Hearing	2.50	0.93	8
	Pop	Deaf	2.56	0.88	9
		Hearing	2.75	0.78	8
	Rap	Deaf	2.22	0.97	9
		Hearing	2.88	0.35	8

	Rock	Deaf	2.78	0.67	9
		Hearing	2.25	1.04	8

Table 71: Descriptive statistics for the levels of agreement with statement 2 for each genre and hearing status

Measure	Genre	Hearing Status	Mean	SD	N
The patterns and shapes on the screen were pleasant to watch	Classical	Deaf	1.89	0.93	9
		Hearing	1.25	0.46	8
	Country	Deaf	1.78	0.97	9
		Hearing	1.63	0.74	8
	Jazz	Deaf	1.44	0.88	9
		Hearing	1.75	0.89	8
	Pop	Deaf	1.67	1.00	9
		Hearing	1.25	0.71	8
	Rap	Deaf	1.44	0.88	9
		Hearing	1.25	0.71	8
	Rock	Deaf	1.56	0.88	9
		Hearing	1.63	0.92	8

Table 72: Descriptive statistics for the levels of agreement with statement 3 for each genre and hearing status

Measure	Genre	Hearing Status	Mean	SD	N
The vibrations helped me	Classical	Deaf	1.67	0.87	9

understand the visualization		Hearing	1.88	0.99	8
	Country	Deaf	1.56	0.88	9
		Hearing	1.63	0.92	8
	Jazz	Deaf	1.78	0.97	9
		Hearing	2.00	1.07	8
	Pop	Deaf	1.44	0.88	9
		Hearing	1.75	0.89	8
	Rap	Deaf	1.67	1.00	9
		Hearing	1.38	0.74	8
	Rock	Deaf	1.67	1.00	9
		Hearing	2.13	0.99	8

Table 73: Descriptive statistics for the levels of agreement with statement 4 for each genre and hearing status

Measure	Genre	Hearing Status	Mean	SD	N
The visualization helped me understand the vibrations	Classical	Deaf	1.56	0.88	9
		Hearing	1.88	0.99	8
	Country	Deaf	1.67	0.87	9
		Hearing	1.75	0.89	8
	Jazz	Deaf	1.44	0.88	9
		Hearing	2.00	1.07	8
	Pop	Deaf	1.67	1.00	9
		Hearing	1.38	0.74	8
	Rap	Deaf	1.89	1.05	9

		Hearing	1.75	1.04	8
	Rock	Deaf	1.67	1.00	9
		Hearing	2.13	0.99	8

Table 74: Descriptive statistics for the levels of agreement with statement 5 for each genre and hearing status

Measure	Genre	Hearing Status	Mean	SD	N
The vibrations were too strong	Classical	Deaf	2.67	0.71	9
		Hearing	2.88	0.35	8
	Country	Deaf	2.00	0.87	9
		Hearing	2.50	0.93	8
	Jazz	Deaf	2.56	0.53	9
		Hearing	2.75	0.46	8
	Pop	Deaf	2.78	0.44	9
		Hearing	2.75	0.46	8
	Rap	Deaf	2.33	0.87	9
		Hearing	2.50	0.93	8
	Rock	Deaf	2.67	0.50	9
		Hearing	2.50	0.93	8

Table 75: Descriptive statistics for the levels of agreement with statement 6 for each genre and hearing status

Measure	Genre	Hearing Status	Mean	SD	N

The vibrations up and down my back felt pleasant	Classical	Deaf	1.44	0.73	9
		Hearing	1.50	0.93	8
	Country	Deaf	1.44	0.88	9
		Hearing	1.25	0.71	8
	Deaf	Deaf	1.56	0.88	9
		Hearing	1.38	0.74	8
	Pop	Deaf	1.56	0.88	9
		Hearing	1.13	0.35	8
	Rap	Deaf	1.44	0.88	9
		Hearing	1.25	0.46	8
	Rock	Deaf	1.22	0.67	9
		Hearing	1.38	0.74	8

Table 76: Descriptive statistics for the levels of agreement with statement 7 for each genre and hearing status

Measure	Genre	Hearing Status	Mean	SD	N
The pattern of vibrations on my back was confusing	Classical	Deaf	2.56	0.73	9
		Hearing	2.63	0.74	8
	Country	Deaf	2.78	0.67	9
		Hearing	2.13	0.99	8
	Jazz	Deaf	3.00	0.00	9
		Hearing	2.25	1.04	8
	Pop	Deaf	2.44	0.88	9
		Hearing	2.50	0.93	8

	Rap	Deaf	2.33	1.00	9
		Hearing	2.25	1.04	8
	Rock	Deaf	2.56	0.88	9
		Hearing	2.50	0.93	8

6.15.4.5Within Genres and Between Conditions

Table 77: Descriptive statistics for the levels of agreement with the statement "the movement was on the screen was hard to watch"

Condition		Classical	Country	Jazz	Pop	Rap	Rock
1	Mean	1.60	1.80	2.20	2.20	2.20	2.20
	SD	0.89	1.10	1.10	1.10	1.10	1.10
	N	5	5	5	5	5	5
2	Mean	2.00	2.67	2.75	3.00	2.50	2.50
	SD	1.00	0.58	0.50	0.00	1.00	1.00
	N	3	3	4	3	4	4
3	Mean	2.73	2.45	2.55	2.64	2.64	2.64
	SD	0.65	0.82	0.82	0.81	0.67	0.81
	N	11	11	11	11	11	11

Table 78: Descriptive statistics for the levels of agreement with the statement “the patterns and shapes on the screen were pleasant to view”

Condition		Classical	Country	Jazz	Pop	RHRB	Rock
1	N	5	5	5	5	5	5
	Mean	1.40	2.00	2.40	2.20	1.80	2.00

	SD	0.55	1.00	0.89	1.10	1.10	1.00
2	N	3	3	4	3	4	4
	Mean	2.33	1.67	1.00	1.00	1.00	1.50
	SD	0.58	0.58	0.00	0.00	0.00	1.00
3	N	11	11	11	11	11	11
	Mean	1.45	1.73	1.45	1.36	1.36	1.45
	SD	0.82	0.91	0.82	0.81	0.81	0.82

Table 79: Descriptive statistics for the levels of agreement with the statement "the vibrations helped me understand the visualization"

Condition		Classical	Country	Jazz	Pop	RHRB	Rock
1	N	4	4	5	5	5	5
	Mean	2.75	1.50	2.60	1.80	1.80	2.40
	SD	0.50	1.00	0.89	1.10	1.10	0.89
2	N	3	3	4	3	4	4
	Mean	2.00	1.33	1.50	1.33	1.50	1.75
	SD	1.00	0.58	1.00	0.58	1.00	0.98
3	N	11	11	11	11	11	11
	Mean	1.27	1.64	1.64	1.45	1.27	1.55
	SD	0.65	0.92	0.92	0.82	0.65	0.93

Table 80: Descriptive statistics for the levels of agreement with the statement “the visualization helped me understand the vibrations”

Condition		Classical	Country	Jazz	Pop	RHRB	Rock
1	N	4	4	5	5	5	5

	Mean	2.75	2.00	2.20	1.80	2.20	2.40
	SD	0.50	1.16	1.10	1.10	1.10	0.89
2	N	3	3	4	3	4	4
	Mean	2.33	1.67	1.00	1.33	1.50	1.75
	SD	0.58	0.58	0.00	0.58	1.00	0.96
3	N	11	11	11	11	11	11
	Mean	1.18	1.64	1.55	1.45	1.55	1.55
	SD	0.60	0.81	0.93	0.82	0.93	0.93

Table 81: Descriptive statistics for the levels of agreement with the statement: “the vibrations were too strong"

Condition		Classical	Country	Jazz	Pop	RHRB	Rock
1	N	4	4	5	5	5	5
	Mean	2.50	2.75	2.40	2.40	2.40	2.80
	SD	0.58	0.50	0.89	0.89	0.89	0.45
2	N	3	4	4	4	4	4
	Mean	3.00	2.50	2.50	3.00	2.00	1.75
	SD	0.00	0.58	0.58	0.00	1.16	0.98
3	N	11	11	11	11	11	11
	Mean	2.82	2.00	2.64	2.73	2.36	2.55
	SD	0.60	1.00	0.51	0.47	0.92	0.82

Table 82: Descriptive statistics for the levels of agreement with the statement "the vibrations up and down my back felt pleasant"

Condition	Classical	Country	Jazz	Pop	RHRB	Rock

1	N	4	4	5	5	5	5
	Mean	2.25	1.00	1.80	1.40	1.60	1.40
	SD	0.96	0.00	1.10	0.89	0.89	0.89
2	N	4	4	4	4	4	4
	Mean	1.50	1.25	1.00	1.25	1.50	1.00
	SD	1.00	0.50	0.00	0.50	1.00	0.00
3	N	11	11	11	11	11	11
	Mean	1.27	1.55	1.55	1.45	1.27	1.27
	SD	0.65	0.93	0.82	0.82	0.65	0.65

Table 83: Descriptive statistics for the levels of agreement with the statement "the pattern of vibrations on my back was confusing"

Condition		Classical	Country	Jazz	Pop	RHRB	Rock
1	N	4	4	5	5	5	5
	Mean	2.25	3.00	2.60	2.40	2.20	2.60
	SD	0.96	0.00	0.89	0.89	1.10	0.89
2	N	4	4	4	4	4	4
	Mean	2.50	3.00	3.00	2.00	2.50	3.00
	SD	0.58	0.00	0.00	1.16	1.00	0.00
3	N	11	11	11	11	11	11
	Mean	2.73	2.18	2.64	2.73	2.27	2.45
	SD	0.65	0.98	0.81	0.65	1.01	0.93

6.16 Not Significant Post-study Results

6.16.1 Levels of Enjoyability

6.16.1.1Between Conditions

Table 84: Descriptive statistics for the levels of enjoyability for each condition

Condition	Mean	N	SD
1	3.00	5	0.00
2	3.00	4	0.00
3	2.73	11	0.65
Total	2.85	20	0.49

6.16.1.2 Between Hearing and Deaf Groups

Table 85: Descriptive statistics for the levels of enjoyability for each hearing status

Hearing Status	Mean	N	SD
Deaf	2.70	10	0.68
Hearing	3.00	10	0.00
Total	2.85	20	0.49

6.16.2 Levels of Agreement

6.16.2.1Between Hearing and Deaf Groups

Table 86: Descriptive statistics for the levels of agreement of each statement for each hearing status

Hearing Status	Statement 1	Statement 2	Statement 3	Statement 4

Deaf	Mean	1.40	1.70	2.40	1.30
	N	10	10	10	10
	SD	0.84	0.95	0.97	0.68
Hearing	Mean	1.00	1.50	2.80	1.10
	N	10	10	10	10
	SD	0.00	0.85	0.63	0.32
Total	Mean	1.20	1.60	2.60	1.20
	N	20	20	20	20
	SD	0.62	0.88	0.82	0.52

References

[1] A. Alexandrakis, "The role of music and dance in ancient greek and chinese rituals: form versus content," *Journal of Chinese Philosophy,* vol. 33, pp. 267-278, 2006.

[2] G. Ilie and W.F. Thompson, "A comparison of acoustic cues in music and speech for three dimensions of affect," *Music Perception: An Interdisciplinary Journal,* vol. 23, pp. 319, 2006.

[3] D.W. Fourney and D.I. Fels, "Creating access to music through visualization," in *Proceedings of IEEE International Conference Science and Technology for Humanity*, 2009, pp. 939-944

[4] P.G. Hunter and E.G. Schellenberg, "Music and emotion," in *Music Perception*, New York, Springer, 2010, pp. 129-164.

[5] H. Bringman et al., "Relaxing music as pre-medication before surgery: a randomised controlled trial," *Acta Anaesthesiologica Scandinavica,* vol. 53, pp. 759-764, 2009.

[6] M. Eldridge et al., "Seeing what you hear: visual feedback improves pitch recognition," *European Journal of Cognitive Psychology,* vol. 22, pp. 1078-1091, 2010.

[7] D. Fourney and D. Fels, "“Thanks for pointing that out.” making sarcasm accessible for all," in *Proceedings of the Human Factors and Ergonomics Society,* vol. 52, 2008, pp. 571-575.

[8] M. R. Leek et al., "Enjoyment of music by elderly hearing-impaired listeners," *Journal of the American Academy of Audiology,* vol. 19, pp. 519-526, 2008.

[9] M. J. Middelweerd and R. Plomp, "The effect of speech reading on the speech-reception threshold of sentences in noise," *Journal of the Acoustic Society of America,* vol. 82, pp. 2145-2147, 1987.

[10] M. Pouris and D. Fels, "Creating an entertaining and informative music visualization," in *Proceedings of 13th International Conference of Computers Helping People*, vol. 7382, 2012, pp. 451-458.

[11] R. Hiraga et al., "Music learning through visualization," in *Proceedings of Second International Conference of Web Delivering Music*, 2002, pp. 101-108.

[12] T. Kunze and H. Taube, "See: a structured event editor - visualizing compositional data in common music," in *Proceedings of the International Computer Music Conference*, 1996, pp. 7-10.

[13] S.M. Smith and G.N. Williams, "A visualization of music," in *Proceedings of Visualization '97.*, 1997, pp. 499-503.

[14] L.E. Marks, "On cross-modal similarity: the perceptual structure of pitch, loudness, and brightness," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 15, pp. 586-602, 1989.

[15] S. Nanayakkara et al., "An enhanced musical experience for the deaf: design and evaluation of a music display and a haptic chair," in *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, 2009, pp. 337-346.

[16] M. Zentner et al., "Emotions evoked by the sound of music: characterization, classification, and measurement," *Emotion*, vol. 8, pp. 494-521, 2008.

[17] M. Friendly, "A brief history of data visualization," in *Handbook of Data Visualization*, 2008, pp.15-56.

[18] R. Kosara, "Visualization criticism - the missing link between information visualization and art," in *Proceedings of International Conference on Information Visualization*, 2007, pp. 631-636.

[19] D. A. Keim, "Information visualization and visual data mining," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, pp. 1-8, 2002.

[20] G. G Robertson et al., “Information visualization using 3d interactive animation,” *Communications of the ACM,* vol. 36, pp. 57-71, 1993.

[21] J. A. Sloboda, "Empirical studies of emotional response to music," 1992, pp. 33-46.

[22] P. Bach-y-Rita and S. W. Kercel, "Sensory substitution and the human–machine interface," *Journal of Trends Cognitive. Sciences,* vol. 7, pp. 541-546, 2003.

[23] N. Kosugi, "Misual: music visualization based on acoustic data," in *Information Integration and Web-based Applications and Services,* 2010, pp. 609-616.

[24] M. Cooper et al., "Visualization in audio-based music information retrieval," *Computer Music Journal,* vol. 30, pp. 42-62, 2006.

[25] A. S. Al-Hammadi and R. H. Milne, "A neuro-fuzzy classification approach to the assessment of student performance," in *Proceedings of IEEE International Conference on Fuzzy Systems,* vol. 2, pp. 837-841, 2004.

[26] R. Hiraga and N. Matsuda, "Visualization of music performance as an aid to listener's comprehension," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, 2004, pp. 103-106.

[27] B. Caterina et al., "Independent mechanisms for ventriloquism and multisensory integration as revealed by theta-burst stimulation," *European Journal of Neuroscience,* vol. 31, pp. 1791-1799, 2010.

[28] M. Laurson et al., "The visualization of computer-assisted music analysis information in pwgl," *Journal of New Music Research,* vol. 37, pp. 61-76, 2008.

[29] M. Laurson and M. Kuuskankare, "pwgl: a novel visual language based on common lisp, clos and opengl," in *Proceedings of International Computer Music Conference*, pp. 142-145, 2002.

[30] R. Brown, "Audio activated video display," U.S Patent 4 081 829, August 23, 1976.

[31] AtariVideoMusic. *Video Music* [Online]. Available: http://www.atarihq.com/dedicated/videomusic.php

[32] J. B. Mitroo et al., "Movies from music: visualizing musical compositions," in *Conference on Computer Graphics and Interactive Techniques*, vol. 13, pp. 218-225, August, 1979.

[33] D. North. (2007, December 26). PS3's music visualizer gaia explained [Online]. Available: http://www.destructoid.com/ps3-s-music-visualizer-gaia-explained-61433.phtml

[34] C. Grant. (2005, November 18). Visualize this: xbox 360's visualizer taken for a spin [Online]. Available: http://www.joystiq.com/2005/11/18/visualize-this-xbox-360s-visualizer-taken-for-a-spin/

[35] P. Strauss. (2007, August 24). Atari video music: forgotten 1970s tech [Online]. Available: http://technabob.com/blog/2007/08/24/atari-video-music-forgotten-1970s-tech/

[36] Microsoft. Windows media player 12 [Online]. Available: http://windows.microsoft.com/en-us/windows/windows-media-player

[37] Apple. iTunes 11 [Online]. Available: http://www.apple.com/itunes/

[38] WinAmp. WinAmp media player [Online]. Available: http://www.winamp.com/

[39] Sony. PlayStation 3 [Online]. Available: http://us.playstation.com/ps3/

[40] Microsoft. Xbox 360 [Online]. Available: http://www.xbox.com/en-CA/Xbox360?xr=shellnav

[41] B. Sheffield. (2007, December 26). Special: q-games on ps3's gaia music visualizer [Online]. Available: http://www.gamasutra.com/php-bin/news_index.php?story=16748.

[42] X. Hua et al., "Automatic music video generation based on temporal pattern analysis," in *Proceedings of the 12th Annual ACM International Conference on Multimedia*, 2004, pp. 472-475.

[43] S. Malinowski. Music animation machine [Online]. Available: http://www.musanim.com/

[44] S. Malinowski. Music animation machine user guide [Online]. Available: http://www.musanim.com/player/MAMPlayerUserGuide.pdf

[45] S. Malinowski. Music animation machine midi player [Online]. Available: http://www.musanim.com/player/

[46] P. Ciuha, et al., "Visualization of concurrent tones in music with colours," in *International Conference on Multimedia*, 2010, pp. 1677-1680.

[47] A. Ortony and T. J. Turner, "What's basic about basic emotions?," *Psychological Review*, vol. 97, pp. 315-331, 1990.

[48] W. Schiff et al., "Persistent fear responses in rhesus monkeys to the optical stimulus of looming," *Science*, vol. 136, pp. 982-983, 1962.

[49] E. Schubert, "Modeling perceived emotion with continuous musical features," *Music Perception*, vol. 21, pp. 561-585, 2004.

[50] D. Loeffler, "Instrument timbres and pitch estimation in polyphonic music," 2006.

[51] I. Peretz and K. L. Hyde, "What is specific to music processing? Insights from congenital amusia," *Trends Cognitive Science*, vol. 7, pp. 362-367, 2003.

[52] J. M. Geringer and M. D. Worthy, "Effects of tone-quality Changes on intonation and tone-quality ratings of high school and college instrumentalists," *Journal of Research in Music Education*, vol. 47, pp. 135-149, 1999.

[53] R. Gault, "Progress in experiments on tactual interpretation of oral speech," *Journal of Abnormal Psychology and Social Psychology*, vol. 19, pp. 155-159, 1924.

[54] S. Alcorn, "The tadoma method," vol. 34, pp. 195-198, 1932.

[55] C. Branje, et al., "Vibrotactile display of music on the human back," in *Third International Conference on Advances in Computer-Human Interactions*, 2010, pp. 154-159.

[56] M. Rothenberg and R. T. Verrillo, "Vibrotactile frequency for encoding speech parameters," *Journal of the Acoustical Society of America,* vol. 59, pp. 69-69, 1976.

[57] H. Pongrac, "Vibrotactile perception: examining the coding of vibrations and the just noticeable difference under various conditions," *Multimedia Systems,* vol. 13, pp. 297-307, 2008.

[58] G. D. Goff, "Differential discrimination of frequency of cutaneous mechanical vibration," *Journal of Experimental Psychology,* vol. 74, pp. 294-299, 1967.

[59] D. A. Mahns,et al., "Vibrotactile frequency discrimination in human hairy skin," *Journal of Neurophysiology,* vol. 95, pp. 1442-1450, 2006.

[60] K. A. Kaczmarek et al., "Electrotactile and vibrotactile displays for sensory substitution systems," *IEEE Transactions on Biomedical Engineering,* vol. 38, pp. 1-16, 1991.

[61] A. G. De Volder et al., "Changes in occipital cortex activity in early blind humans using a sensory substitution device," *Brain Research,* vol. 826, pp. 128-134, 1999.

[62] J. P. Rauschecker, "Compensatory plasticity and sensory substitution in the cerebral cortex," *Trends in Neuroscience,* vol. 18, pp. 36-43, 1995.

[63] H. Z. Tan and A. Pentland, "Tactual displays for sensory substitution and wearable computers," in *ACM SIGGRAPH Courses,* 2005.

[64] M. Kitagawa et al., "Effect of sensory substitution on suture-manipulation forces for robotic surgical systems," *Journal of Thoracic Cardiovascular Surgury,* vol. 129, pp. 151-158, 2005.

[65] P. Richard and P. Coiffet, "Human perceptual issues in virtual environments: sensory substitution and information redundancy," in *4th IEEE International Workshop on Robot and Human Communication,* 1995, pp. 301-306.

[66] M. Karam, et al., "Modelling perceptual elements of music in a vibrotactile display for deaf users: a field study," in *2nd International Conference on Advances in Computer-Human Interactions,* 2009, pp. 249-254.

[67] M. Karam and D. I. Fels, "Designing a model human cochlea: issues and challenges in cross-modal audio-haptic displays," in *Proceedings of the 2008 Ambi-Sys Workshop on Haptic User Interfaces in Ambient Media Systems*, 2008, pp. 1-9.

[68] M. Karam, et al., "The emoti-chair: an interactive tactile music exhibit," in *Proceedings of the 28th of the International Conference Extended Abstracts on Human Factors in Computing Systems*, 2010, pp. 3069-3074.

[69] D. De Ridder et al., "Functional anatomy of the human cochlear nerve and its role in microvascular decompressions for tinnitus," *Neurosurgery*, vol. 54, pp. 381-390, 2004.

[70] C. Branje, et al., "Enhancing entertainment through a multimodal chair interface," in *International Conference on Science and Technology for Humanity*, 2009, pp. 636-641.

[71] A. Baijal et al., "Composing vibrotactile music: A multi-sensory experience with the emoti-chair," in *IEEE Haptics Symposium*, 2012, pp. 509-515.

[72] C. Branje et al., "The Horror of vibrotactile stimulation,".

[73] N. S. Rickard, "Intense emotional responses to music: a test of the physiological arousal hypothesis," *Psychology of Music*, vol. 32, pp. 371-388, 2004.

[74] S. Khalfa, et al., "Event-related skin conductance responses to musical emotions in humans," *Neuroscience Letters*, vol. 328, pp. 145-149, 2002.

[75] Jogamp. JOGL project [Online]. Available: http://jogamp.org/jogl/www/

[76] Midi, General midi standard [Online]. Available: http://www.midi.org/techspecs/gm.php;

[77] Midi. Summary of midi messages [Online]. Available: http://www.midi.org/techspecs/midimessages.php

[78] Apple. Concurrency and opengl [Online]. Available: https://developer.apple.com/library/mac/#documentation/graphicsimaging/conceptual/OpenGL-

MacProgGuide/opengl_threading/opengl_threading.html#//apple_ref/doc/uid/TP40001987-CH409-SW1

[79] J. Jacoby and M. S. Matell, "Three-point likert scales are good enough," *Journal Market Research* vol. 8, pp. 495-500, 1971.

[80] A. Mehrabian and J. A. Russell, *An Approach to Environmental Psychology.* Cambridge, MA, US: The MIT Press, 1974.

[81] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): results of empirical and theoretical research," *Human Mental Workload,* vol. 1, pp. 139-183, 1988.

[82] G. Schwartz and D. Weinberger, "Patterns of emotional responses to affective situations: Relations among happiness, sadness, anger, fear, depression, and anxiety," *Motivation and Emotion,* vol. 4, pp. 175-191, 1980.

[83] A. L. Riley et al., "The role of endorphins in animal learning and behavior," *Neuroscience & Biobehavioral Reviews,* vol. 4, pp. 69-76, 1980.

[84] S. V. Kasl, "Stress and Health," *Annual Review of Public Health,* vol. 5, pp. 319-341, 1984.

[85] Midi. Summary of pitch bend messages [Online]. Available: http://www.indiana.edu/~emusic/etext/MIDI/chapter3_MIDI4.shtml

[86] Oracle. Java api 6 midi package [Online]. Available: http://docs.oracle.com/javase/6/docs/api/javax/sound/midi/package-frame.html

[87] Midi. Midi percussion messages [Online]. Available: http://www.midi.org/techspecs/gm1sound.php

[88] OpenGL. OpenGL and multi-threading [Online]. Available: http://www.opengl.org/wiki/OpenGL_and_multithreading