Generating a Genre: An Algorithmic Approach to Creating Popular Music

by

Paul B. Maurer

Submitted in partial fulfillment of the requirements for the
Master of Music in Music Technology
in the Department of Music and Performing Arts Professions
in The Steinhardt School
New York University
Advisor: Dr. Morwaread M. Farbood
[DATE: 2009/06/17]

Abstract

The compositional rules of musical genres can be entered into a computer program that reproduces music of the same genre. Such a program is useful as a tool for compositional assistance and other genre-specific musical needs. A large selection of the most popular songs in the contemporary urban genre is analyzed and the musical components are converted into statistics for stochastic algorithms. A Max/MSP program uses these algorithms to generate phrases of music for this genre. The musical output of the program is accurate and identified as contemporary urban music.

Acknowledgments

I would like to thank Dr. Kenneth Peacock and Dr. Robert Rowe for their continued support throughout my time in the Music Technology program at New York University, and Dr. Morwaread Farbood and Dr. Juan Bello for being my thesis advisors and guiding me through the entire process.

I would also like to thank my parents, Keith and Judith Maurer, for their unending support for me throughout my entire degree and my entire life, and to Sarah Würzburger, for her support and indulgence while I worked on this thesis in so much of my spare time, and her invaluable feedback about all of the work.

In addition, I would like to thank the following people:

- Phillip Manget, for introducing me to Max/MSP and algorithmic composition, and for countless conversations about how to "take everything to the next level"
- Benjamin Hendricks, for always being willing to help me with feedback and ideas, including crazy ideas that only the two of us would appreciate
- Robert Linke, for all of the discussions about the Max/MSP program
- Ema Jolly, for her excellent musical feedback about the results of the program
- Dr. Steven Rosenhaus, for introducing me to the other side of composition to which I had never been exposed
- Langdon Crawford, for both his support in the Music Technology program and our musical collaborations
- Klara Palotai, for her wonderful hospitality while I finished my degree

Finally, I want to thank Native Instruments for giving me the opportunity to complete my degree during the summer, as well as for making such wonderful software.

Contents

Abstract **2**

Acknowledgments **3**

Preface **5**

1 Introduction **7**

1.1 Motivation 7
1.2 What the Program IS 9
1.3 What the Program IS NOT 9
1.4 Brief History 10
1.5 Related Products 12

2 Data Analysis **17**

2.1 Song Selection 17
2.2 Analysis Components 17
2.3 Data Significance 22

3 The Program **30**

3.1 General Overview 30
3.2 Musical Elements 32

4 Results **39**

4.1 Does It Work? 39
4.2 Conclusion 43
4.3 Future Work 44

5 Bibliography **46**

6 Appendices **48**

Appendix A – List of Analyzed Songs 48
Appendix B – Max/MSP Program Views 49

Preface

In this paper, I discuss how I develop a program that algorithmically generates compositional ideas for a specific genre of music: contemporary urban. In order to accomplish this, I created a program using Max/MSP that can generate this music based on a set of rules that govern the genre, as well as a small set of optional intuitive global parameters that can be set by the user. The music that is generated gives the user ideas for further composition based on the outcome, as well as provides musical phrases and loops which could be taken directly from the program and used in other compositions.

Many genres of music are defined by the patterns of the beats, the melodies of the instruments, the instruments chosen, and several other parameters. These parameters, while sometimes very broad, still have a set of "rules" which define the music to be part of a specific genre. They are rules of composition, and ultimately these rules can be defined. A program can use these rules to create music with a perspective that is impartial compared to the human brain, and new original ideas can come from music generated with these same rules. I carefully analyzed the contemporary urban music genre to try to determine what these rules are, and then represented these rules with algorithmic code in the Max/MSP program.

A program of this type allows anyone to create music of a particular genre for whatever purpose he or she might have in mind. The user can enter a few simple parameters of the music that are not specifically related to the genre (optional), and the program takes care of the rest. The program I created generates contemporary urban song ideas "automatically" by using the rules defined in the program. This has several practical uses:

- A user can generate song ideas for a future production, acting as a composition assistant. In a similar manner, it can also help an urban producer who is having writer's block.
- Temporary urban tracks can be created for media such as advertisements, film, video games, and websites.
- An urban vocalist can use the program to generate unique songs over which he or she can practice singing or rapping. It can also be used to practice improvisational techniques.
- Audio loops can be instantly created and recorded for loop-based compositions, loop slicing and other audio production uses.

To obtain the rules of composition for contemporary urban music, the music must be thoroughly analyzed. This was accomplished by selecting the top 50 contemporary urban music singles from the current musical charts. Each song was broken down into its individual components for analysis. All data was analyzed by ear and from this data, the basic rules of composition were derived and entered into the program.

The program has one interface view; the main portion of the interface is for generating new music and individual phrases, and the secondary portion includes only the basic parameters common to most styles of music, allowing the user to match a basic sound for which they want to use the music. The program generates the music with the internal rules already in place by the program. The data generated by the program is in the form of MIDI files that are loaded into VST instrument plugins, using both sample and synth based sounds. The output is musical phrases of the contemporary urban genre.

1 Introduction

1.1 Motivation

Before I started studying the 'ins' and 'outs' of urban music, I never really paid attention to the differences, subtle or not, among the varying styles within that genre. As I began to approach the Max/MSP program discussed in this paper, I really had to think about what sub-genre I would focus on. To be precise, the specific genre title of the music I chose to generate would be "contemporary western urban pop music", for the following reasons. The global genre is considered "urban" because it contains elements from Hip-Hop, R&B, and many associated sub-genres. The music is "contemporary" because I analyzed urban music that was popular within the last couple of years. The word "pop" in the title reflects two things: the music is "popular" because the songs were listed in the top rankings of the urban music charts, and they relate to "pop" music as they are generally catchy and repetitive. It is considered "western" because the music was selected from American and European music charts. As for the term "music", if you were to ask my parents I am sure that the validity of this term would be open to debate. But for the purposes of this paper, I will usually refer to the music simply as "urban".

What really defines a genre of music? If you were to ask someone about a song that is playing on the radio, "What genre of music is this?" he or she would probably be able to give you a general answer that would be satisfactory. But then if you were to ask, "How do you know?" the answer would not come as easily. You would probably get an answer that contains general characteristics of the music, such as the types of instruments used or the style of the vocal line. Often those same characteristics could be used to describe other genres of music that could be considerably different. Of course, it is

possible to be specific enough to answer that question too, but what about a detailed definition of what makes a *good* song in a specific genre of music, relating to rhythmic structure, melodic patterns, and combined instrumentation? Here is where it gets tricky. A composer of that genre may know the answer, but even then, the answer may come more from the "gut" rather than a true detailed definition of the genre. I am not claiming that it is necessary to define a genre so descriptively, but rather than an answer does exist and that a genre does indeed have rules which can be defined, and that innovation can be achieved by exploiting the varieties of music that can be created by staying within the confines of these rules.

To show that this is true, I analyzed the top 50 contemporary urban songs and collected all of the relevant song data that I could, such as the types of sounds used, the rhythmic patterns of the beats and melodies, the progression of the melodic and harmonic lines, and many other components. I converted all of this data into statistics for calculating probability algorithms. Using Max/MSP, I created a program that brought all of these sounds and probability algorithms together, and it creates its own musical phrases based on the data collected from the original 50 songs.

I chose urban music for this program because while it can be very diverse (as opposed to a more rigid style such as techno music), it still has certain patterns and boundaries which can be realized with enough exposure (as opposed to an extremely diverse style such as jazz music). This allows for many innovative possibilities while still being able to define boundaries without a daunting number of rules. Another reason I chose this genre was that it mostly involves electronic instruments, whether that means they are sampled drums or instruments, or they are synthesized sounds. This allows the

program to generate songs that can potentially have the same sound as the ones that already exist.

1.2 What the Program IS

The program is a tool for generating phrases of music that are based on the contemporary urban genre. It creates music that is recognized as this genre and no other. It exists to show that when all musical components of a consistent-genred body of music are analyzed, rules can be formulated from the data. This particular program uses probability algorithms based on the statistical figures derived from the data collection. The program's output is both wave audio and MIDI files of the generated music.

1.3 What the Program IS NOT

The program is not meant to be a "hit maker", producing one great popular song after the other, despite the fact that the analyzed songs come from the top of the urban charts. The program only creates music that is recognizable as the contemporary urban genre, and it can be used as a tool for generating new ideas. While it is technically possible that the program generates a musical phrase that could be turned into a hit song due to the random nature of the program, it is certainly not intended to do so.

The urban genre, along with several other genres, is very strongly influenced by the vocals. While it is conceivable to create an additional program to generate *lyrics*, or an additional algorithm that generates a basic vocal *melody*, it is not conceivable to create one that generates vocal *lines* (at least not useful ones) with current technology and available programming resources. Therefore, this program is not a generator of vocal

lines. It is better to think of the music generated by the program as the main influence for the vocals, which would be added later.

The program is also not a stand-alone application as the program uses several commercial virtual instrument plugins by Native Instruments, with Max/MSP as the host. Instead, this program functions as the model for a program that could potentially be developed without the use of existing commercial software.

1.4 Brief History

The history of algorithmic composition is so extensive that a full thesis could be written about just one aspect of it. Only a few representative cases are mentioned here because they are directly related to the work in this study.

Algorithmic composition has been around for a very long time, with the first musical application dating at around AD 1000 (Nierhaus, 2009). Guido of Arezzo developed a system for composition using text, where the consonants and vowels, syllables, and pauses were mapped to pitches and melodic phrases (Nierhaus, 2009). One of the first uses of algorithmic composition that had a widespread historical significance was a game developed in the eighteenth century called *Musikalisches Würfelspiel*, typically involving a pair of dice and a matrix with references to several measures of musical passages (Cope, 2001). Each game had practically unlimited possibilities and was used by several well-known composers of the time such as Mozart, Haydn, and C.P.E. Bach, among others (Cope, 2001).

Computer assisted algorithmic composition dates back to 1955 with a composition called the “Illiac Suite” by Lejaren Hiller and Leonard Isaacson (Bohn).

The ILLIAC I computer assisted in the composition of the piece and then it was interpreted for use with a string quartet (Nierhaus, 2009). In 1971, Iannis Xenakis used several computational models in his algorithmic works, such as Markov chains, probability laws and stochastics (Cope, 2001). His algorithmic works are often combined with his own compositions, defining his style (Cope, 2001). Xenakis is widely considered as the father of computer based composition (Cope, 2001).

David Cope developed a program called "Experiments in Musical Intelligence", or EMI, which uses a database of works by other composers to generate new works that have the same style as the original composers (Cope, 2001, 2005). EMI uses a data driven model for music composition, meaning that it analyzes the works in a database and then uses this analysis to replicate the new music (Cope, 2005). A user must gather a collection of previously composed works that EMI analyzes for details such as harmonic progressions, polyphony, harmonies, voice ranges, meter, repetition, variation, and many others (Cope, 2005). From these details, new music is produced which attempts to emulate the style of the original composer (Cope, 2005).

While computer based composition is a relatively new concept, rule-based composition is not new at all. Most of the classic genres of music were based on rather strict rules for composition, influencing melodies, harmonies, and rhythmic patterns. In fact, so much of music theory is based on rules that one could almost say that all music originated from some form of rule-based composition.

1.5 Related Products

There are other products on the market that create a form of mainstream music by using algorithmic composition. Microsoft's "Songsmith" for example is an interactive program that analyzes the input of a user's voice and generates a song that accompanies the vocal frequencies (Microsoft Research, 2009). The generated song uses a set of chord progression rules, along with the basic style and instrumentation of a selection of music genres (Microsoft Research, 2009). It forms the structure of the song based on the timing of the vocal input and a basic general song structure algorithm (Microsoft Research, 2009). Often termed as "reverse karaoke", Songsmith is best used as a tool to sing an idea into the computer and have a basic song structure form around it, acting as a "musical sketchbook" (Sadun, 2009). The generated songs are very basic and do not go much beyond a simple standard structure combining basic elements of composition and musical theory (Sadun, 2009). The program does not "innovate", as it is strongly rooted to conventional composition rules with pre-programmed structural guidelines.

Another related program is called "Band-In-A-Box" by PG Music. Band-In-A-Box requires the user to input chord progressions and to choose a style of music, and then the program generates a full song arrangement based on that input (PG Music, 2009). The songs are generated by loading patterns and chord & melodic progressions from a very large database of pre-programmed sections, depending on the style of music chosen (PG Music, 2009). Most basic elements of the songs are also editable, such as instrumentation, tempo, time signature, swing, etc. (PG Music, 2009). There is more emphasis on the overall song formation with genuine styles of playing compared to

Songsmith, which is focused more on a basic structure surrounding a vocal line. See Figure 1.1 for a typical view of the Band-In-A-Box interface (PG Music, 2009).

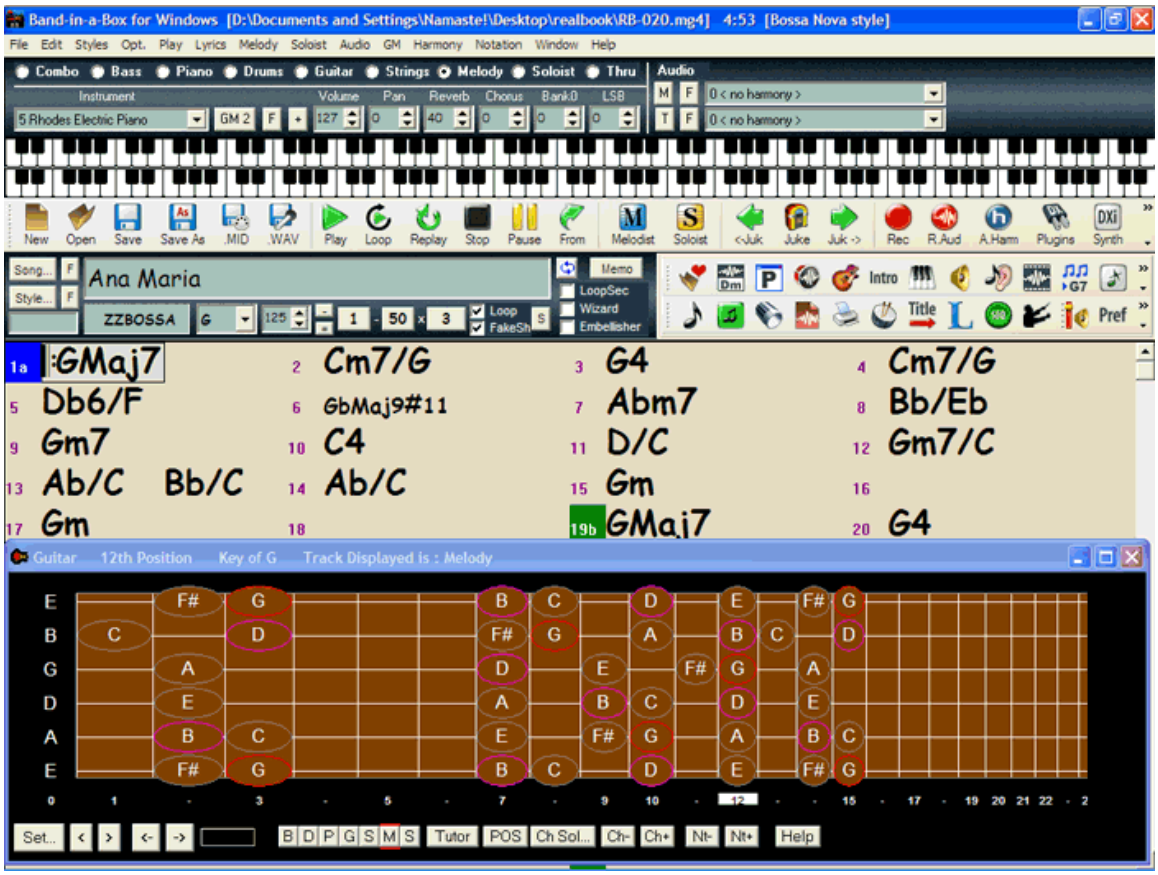


Figure 1.1: Band-In-A-Box interface.

A program called "Jammer" by SoundTrek is similar to Band-In-A-Box in that is creates a full song based on user input such as chord progressions and musical style (SoundTrek, 2009). With Jammer however, a user can generate new changes in the song for individual measures, single instruments and drums, or the entire song (SoundTrek, 2009).

"Jamstix" is a program by Rayzoon Technologies that uses a strictly algorithmic approach to creating drumbeats (Rayzoon, 2009). It does not recall static patterns from a database, but rather uses a real-time groove algorithm, which is based on drummer playing style modeling (Rayzoon, 2009). The idea is to have a realistic sounding drum track that could be played by a human drummer and is analogous to having a live drummer to play along with for jamming or recording (Rayzoon, 2009). It randomizes several sound aspects of the individual drums to avoid the sound of computerized repetition (also known as the "machine gun effect") (Rayzoon, 2009). The focus of the program is to generate the most human sounding acoustic drummer to play along to without having an actual drummer available. Figure 1.2 shows one of the main interface pages for Jamstix 2 (Rayzoon, 2009).

Figure 1.2: Jamstix 2 interface.

Finally, another unique generative program is called “Muse” by the company Abaltat (Abaltat, 2009). The idea behind this program is to generate music for video based on the visual properties of the video at a specific time (Abaltat, 2009). The program uses a video recognition algorithm to analyze properties such as color, clip duration and keyframe events in combination with basic musical rules entered by the user (Abaltat, 2009). The rules for the generated music are based on genre algorithms that each has its own selection of instruments, rhythmic patterns and melodic lines (Abaltat, 2009). It also uses an algorithmic approach to composition, as opposed to a database of set patterns. Figure 1.3 gives a basic overview of the Muse functions (Abaltat, 2009).

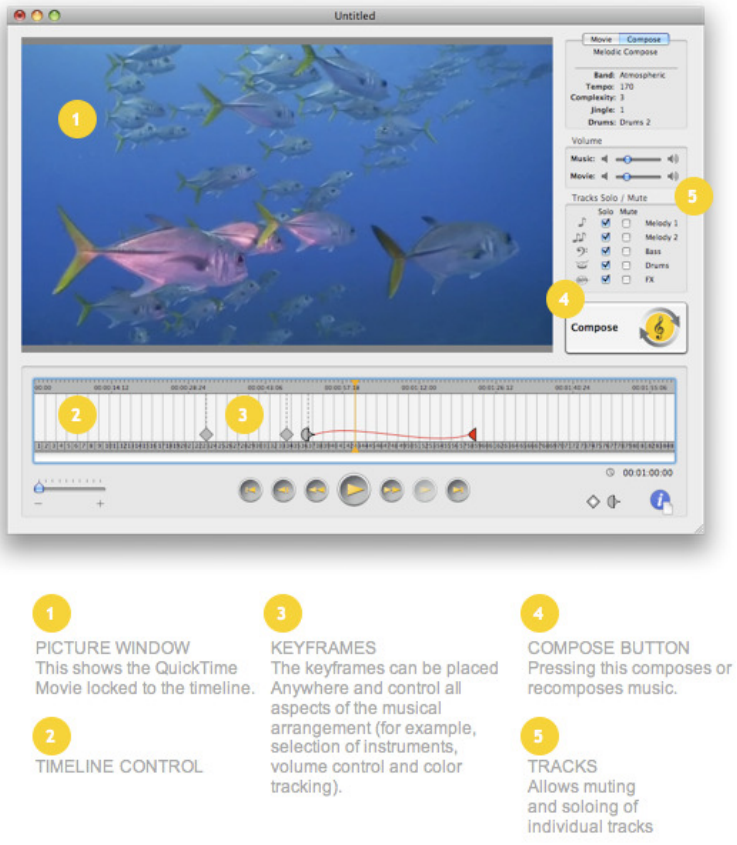


Figure 1.3: Muse interface with functional overview.

The Max/MSP program created for this study is different from the above products for two main reasons. First, except for a few optional non-genre related attributes, the generated music does not depend on user input or external data in order to be created, nor does it use a database of pre-made song components. Instead, it only uses the probability algorithms within the program, designed for the genre. Second, it scrutinizes a particular genre for all relevant musical possibilities in order to recreate that genre's music accurately.

2 Data Analysis

2.1 Song Selection

The selection process for determining which songs to use in the analysis was relatively simple. To find out which contemporary urban songs were the most significant, the Billboard charts were referenced for urban genres of music. Billboard determines the rankings based on several factors such as song sales, digital downloads, radio airplay and internet streams (Contactmusic, 2008). Because the top urban songs are limited to the top 10 or 20, depending on the specific sub-genre (Hip-Hop, R&B, or Rap), the sales rankings of the iTunes store were used for the remaining songs until there was a total of 50 of the top ranking urban songs which could be analyzed. There were more than 50 songs in the top rankings of the charts, but "outliers" were eliminated. Songs were considered outliers if they were made before 2007 (such as Sir Mix-A-Lot's "Baby Got Back"), if they were cover songs (such as Flo Rida's "Right Round"), or if they had song structures that were too far outside the range of the other songs, e.g. a slow R&B song with only a constantly changing piano part and no drums. See Appendix A for a complete list of songs analyzed for this study.

2.2 Analysis Components

Each song was analyzed by ear and all of the data was recorded in the same way for each song. Figure 2.1 shows an example of the song analysis spreadsheet, giving details about the Lil Wayne song "Lollipop".

Song	Artist	Time Sig	Tempo	Key	Scale	Swing
Lollipop	Lil Wayne	4/4	74 BPM	Dm	Minor	Slight

Drums	Kick	Snare	Clap	Snare 2
Sound	Short Analog	Analog	Analog	Analog
Pattern Verse	1-4-7-8-10-11	(none)	5-13	Fills
Pattern Verse alt1	1-6-8-10-11	5-13		
Pattern Chorus	(same as verse)	5-13	5-8-10-13	Fills
Pattern Chorus alt1	(same as verse)	5-8-13		
Additional Effects				
Verse Cycle Length	4 measures	1 measure	1 measure	Fills
Chorus Cycle Length	4 measures	2 measures	1 measure	Fills
Dynmics	Loud	Loud	Medium	Medium
Variations	Break	Break	Break	Fills

Instruments	Bass	Lead	Strings
Sound	Sub Bass	Square Wave	Synth Strings
Style	Sustained	Staccato	Sustained
Polyphony	Solo	Solo	Chords
Pattern Verse	1	all 16th notes	
Pattern Verse alt1	1-13		
Pattern Verse alt2	1-8		
Pattern Chorus	(same as verse)	(same as verse)	1
Pattern Chorus alt1	(same as verse)		
Pattern Chorus alt2	(same as verse)		
Avg Interval Jump	3rd	2nd	3rd
Starting Notes	Tonic	Tonic	Tonic
Ending Notes	Submediant	Dominant	Subdominant
Additional Effects		Reverb	Reverb
Verse Cycle Length	4 measures	2 measures	4 measures
Chorus Cycle Length	4 measures	2 measures	4 measures
Dynmics	Medium	Medium	Quiet
Variations	Break	Break	None

Verse Chord Prog:	1,(3),(7),(6)
Chorus Chord Prog:	1,(3),(7),(4)

Figure 2.1: Analysis of the Lil Wayne song "Lollipop".

Here is a breakdown of each of the analyzed components:

- *Song* – Title of the song
- *Artist* – Performing artist of the song
- *Time Sig* – Time Signature of the song
- *Tempo* – Tempo of the song in Beats Per Minute (BPM)
- *Key* – Key, or principle tonality, of the song

- *Scale* – Scaling used in the song. This is separated from Key because sometimes a song would abstain from using one or more notes in a scale, or would alter the basic scale of the Key in some way.
- *Swing* – General term describing the amount of swing in a song. Swing is the rhythmic shifting of a pattern for a "shuffle" feel. It involves shifting of every second position of a pattern depending on the basic rhythmic unit.
- *Drums* – Lists the types of drum and percussion sounds in the song
- *Instruments* – Lists the types of tonal/pitched instruments used in the song
- *Sound* – Brief general description of the subtype of a drum or instrument
- *Pattern Verse/Chorus* – Rhythmic pattern of the drum or instrument, listing numbers which represent the 16^{th} note divisions of a whole measure
- *Pattern Verse/Chorus alt* - Same as above, but these are the alternate patterns that play during the verse or chorus
- *Additional Effects* – If effects are added to a sound, the type of effect is listed here. This does not include some mastering effects such as compression, unless it is used in a unique or over-processed way.
- *Verse/Chorus Cycle Length* – Length of time (in measures) before the drum or instrument loops its full cycle. In other words, the length in which all main and alternate patterns play before repeating themselves again.
- *Dynamics* – Overall volume of the drum or instrument in relation to the total mix
- *Variations* – If there is some kind of deviation from the main rhythmic or melodic pattern of a drum or instrument, it is listed here.

- *Style* – States if the instrument is staccato, sustained, or decaying (or a combination of these)
- *Polyphony* – States if the instrument plays one note at a time (solo) or more than one note (chords, octaves, etc.)
- *Avg Interval Jump* – Average interval between successive notes in the melody of an instrument
- *Starting/Ending Notes* – Scale degree of the starting/ending note of a complete instrument cycle loop
- *Verse/Chorus Chord Prog* – Chord progression of the verse and chorus, with numbers representing the root position of the chord. Numbers in parenthesis indicate going below the tonic note.

Most of the components above have obvious entry values from their descriptions, but there are a few that need a brief explanation.

The Scale is typically major or minor, but sometimes the song leaves out at least one note in the scale. When this is the case, a string of numbers was entered, each representing the degree in the scale (e.g. '1' represents the tonic, '2' represents the supertonic, etc.). So a Scale value of 1-2-3-4-5-7 leaves out the submediant note in the scale, representing the fact that the note never plays in the song.

The Pattern is always a series of numbers ranging from 1 to 16, where each number represents the rhythmic position of the beginning of the sound. Each measure of every song consists of 4 beats, so the total number of 16^{th} notes in each measure is 16. For example, '1-7-9-13-14-15' would have the rhythm in Figure 2.2.

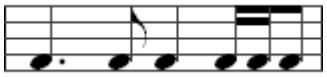


Figure 2.2: Rhythmic notation of the pattern 1-7-9-13-14-15.

The durations of the notes above are arbitrary in this case, as they are calculated separately from the rhythmic position (but are strongly related, as explained in Section 3.2).

The Average Interval Jump is represented by a musical interval value, such as a 3rd or a 5th. Stating whether the interval is major, minor, etc. is not relevant as the notes are always contained within the scale values of the song. The final value adds all of the intervals of each jump contained within a full cycle, and divides that number by the number of jumps in the cycle to get the average interval.

For the Starting/Ending Notes of the instruments, the scale degree that corresponded to the first note and the last note played in a full cycle was recorded. So if a bass line started on the root note of the scale it would be recorded as 'Tonic', and if the last note of the cycle was a third above the root, it would be recorded as 'Mediant'.

The Verse/Chorus Chord Progression numbers show how the chords would change within both the verse and the chorus, with the number values representing the scale position of the root of the chord. Values recorded in parenthesis have chord roots that are below the root of the tonic chord. For example, '1,3,(7),(6)' in the key of C major would have a chord structure like the one in Figure 2.3.

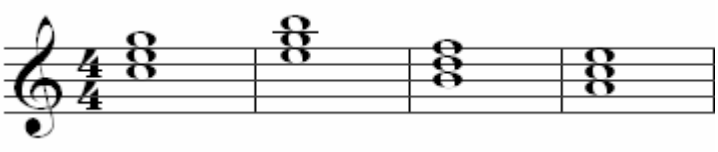


Figure 2.3: Chord progression notation of 1,3,(7),(6).

The chords above are all in the root position; however, they could have originally been in any inversion with the same root note.

The ways that all of these components were incorporated into the program are explained in Section 3.2.

2.3 Data Significance

Most of the gathered data is significant in "typical" ways, which will be shown as the program is described. However, it turns out that not all of the gathered data is significant to song generation, or at least not in a way that could be included in the program. There are obvious elements that can be removed from a more complex calculation, such as the time signature. Every song that was analyzed has a time signature of 4/4 (and this seems to be the case for urban songs in general). So instead of programming a way to change the time signature of the songs, the time signature was simply hard-coded to be 4/4 for every generated song. But some elements are not so obvious, others that are more significant than expected, and still others that are almost counter-intuitive.

The Starting Notes of instrumental parts have a strong significance, with almost every type of instrument having a large portion of their starting notes being on the tonic. Some other instruments (such as the bass) have other starting notes that appear more or

less often than others, revealing the need for these ratios to be calculated into the program. The starting note distribution for the bass is shown in Figure 2.4.

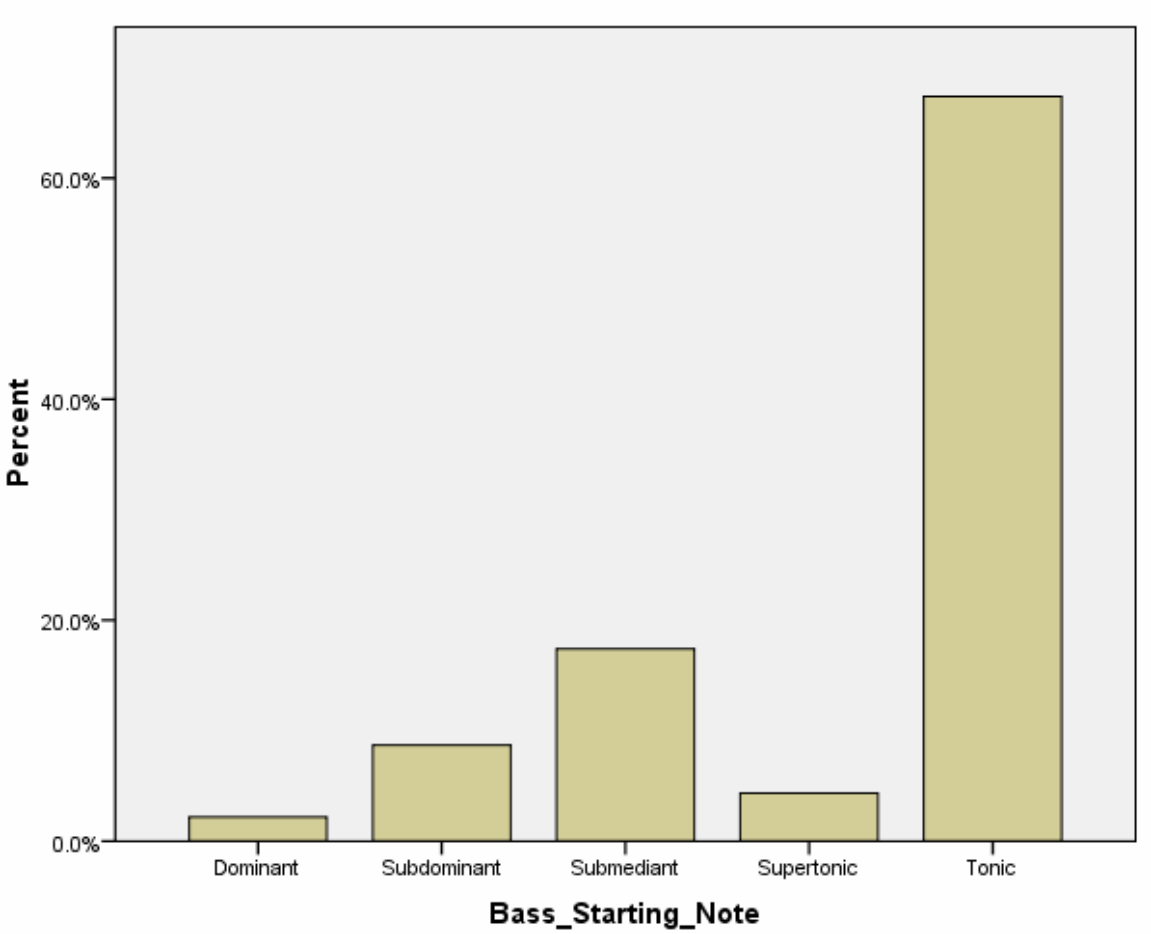


Figure 2.4: Starting note distribution of the bass instrument.

However, the Ending Notes are roughly evenly distributed for all instruments, thus eliminating the need for this value to be a part of the program (beyond giving them an evenly random distribution). Figure 2.5 shows the ending note distribution for the bass, showing a comparatively even distribution.

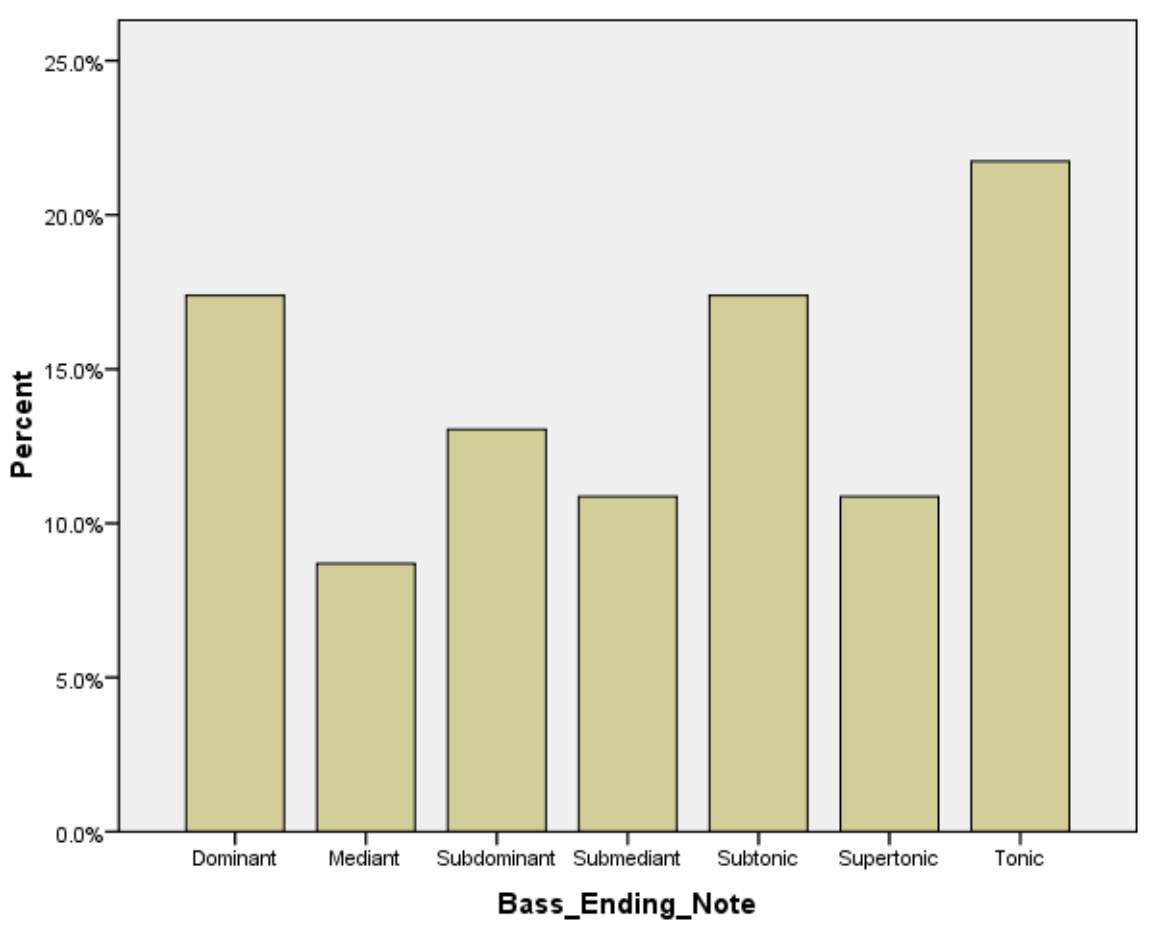


Figure 2.5: Ending note distribution of the bass instrument.

The Chord Progressions of both the verse and the chorus are also not as significant as predicted. The beginning chord and to a lesser degree the second chord are significant, but the remaining chords did not have enough of a distinct recurring pattern for a good algorithm to include with the program. As the Starting Notes mentioned above are significant, the first chord is reflected in that algorithm and no other chord progression programming is necessary.

A correlation analysis was conducted among the drum sounds to see if there was a pattern to the types of drums used (i.e. to see if one type of drum was commonly used when another type was used, or the opposite). The expectation was to find several strong

relationships among the drums, but except for a slight negative correlation between the snare and clap sounds (i.e. when a snare is present, a clap is not, and vice versa) and a slight positive correlation between the tom and crash sounds, none were found (see Figure 2.6).

Correlations

		Kick	Clap	Snare	ClosedHH	OpenHH	Shaker	Crash	Tom	Misc
Kick	Pearson Correlation	.[a]	.[a]	.[a]	.[a]	.[a]	.[a]	.[a]	.[a]	.[a]
	Sig. (2-tailed)		.	.	.	.	.	.	.	.
	N	50	50	50	50	50	50	50	50	50
Clap	Pearson Correlation	.[a]	1	-.434**	.236	.167	.134	.159	.198	.069
	Sig. (2-tailed)	.		.002	.100	.248	.352	.270	.169	.632
	N	50	50	50	50	50	50	50	50	50
Snare	Pearson Correlation	.[a]	-.434**	1	.027	-.064	-.092	-.138	-.187	-.119
	Sig. (2-tailed)	.	.002		.852	.658	.526	.341	.194	.412
	N	50	50	50	50	50	50	50	50	50
ClosedHH	Pearson Correlation	.[a]	.236	.027	1	.116	.232	.249	.177	-.124
	Sig. (2-tailed)	.	.100	.852		.422	.105	.081	.219	.392
	N	50	50	50	50	50	50	50	50	50
OpenHH	Pearson Correlation	.[a]	.167	-.064	.116	1	.214	-.292*	-.208	-.025
	Sig. (2-tailed)	.	.248	.658	.422		.136	.040	.147	.863
	N	50	50	50	50	50	50	50	50	50
Shaker	Pearson Correlation	.[a]	.134	-.092	.232	.214	1	.222	.218	-.127
	Sig. (2-tailed)	.	.352	.526	.105	.136		.122	.128	.381
	N	50	50	50	50	50	50	50	50	50
Crash	Pearson Correlation	.[a]	.159	-.138	.249	-.292*	.222	1	.364**	.033
	Sig. (2-tailed)	.	.270	.341	.081	.040	.122		.009	.818
	N	50	50	50	50	50	50	50	50	50
Tom	Pearson Correlation	.[a]	.198	-.187	.177	-.208	.218	.364**	1	.053
	Sig. (2-tailed)	.	.169	.194	.219	.147	.128	.009		.713
	N	50	50	50	50	50	50	50	50	50
Misc	Pearson Correlation	.[a]	.069	-.119	-.124	-.025	-.127	.033	.053	1
	Sig. (2-tailed)	.	.632	.412	.392	.863	.381	.818	.713	
	N	50	50	50	50	50	50	50	50	50

a. Cannot be computed because at least one of the variables is constant.

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

Figure 2.6: Correlation analysis among the drum types used within the songs.

The same holds true for the instrument sounds. However, when looking for correlations among the drum and instrument sounds together, there was a very strong negative correlation between a sub bass kick drum and a sub bass instrument, meaning that when there was a sub bass kick, there was almost never a sub bass instrument, and vice versa. This makes perfect sense, as the overlapping sounds of a sub bass kick and a sub bass

instrument would cause excessive sub bass to be in the audio mix. Figure 2.7 shows the correlation between the sub bass kick and the sub bass instrument, as well as examples of those sounds compared to others in the same category.

Correlations

		Sub_Bass	Sub_Analog_Kick
Sub_Bass	Pearson Correlation	1.000	-.478**
	Sig. (2-tailed)		.000
	N	50.000	50
Sub_Analog_Kick	Pearson Correlation	-.478**	1.000
	Sig. (2-tailed)	.000	
	N	50	50.000

**. Correlation is significant at the 0.01 level (2-tailed).

Correlations

		Sub_Bass	Punchy_Analog_Kick
Sub_Bass	Pearson Correlation	1.000	.197
	Sig. (2-tailed)		.171
	N	50.000	50
Punchy_Analog_Kick	Pearson Correlation	.197	1.000
	Sig. (2-tailed)	.171	
	N	50	50.000

Correlations

		Synth_Bass	Sub_Analog_Kick
Synth_Bass	Pearson Correlation	1.000	-.010
	Sig. (2-tailed)		.945
	N	50.000	50
Sub_Analog_Kick	Pearson Correlation	-.010	1.000
	Sig. (2-tailed)	.945	
	N	50	50.000

Figure 2.7: The first correlation analysis is between the sub bass and the sub kick sounds, showing a strong negative correlation. The next two show an analysis between a sub bass and a punchy analog kick, and between a synth bass and a sub kick, respectively. These two show no significant correlation.

One surprising attribute, which is much more significant than expected, is the Key attribute. 70% of all analyzed songs are one of the following five keys: C, Db, D, Eb, and E. At the other end of the scale, only 5% of all songs are in one of the keys Bb, F, and G (in fact, none of the songs were in the key of G). See Figure 2.8 to see the distribution of the keys.

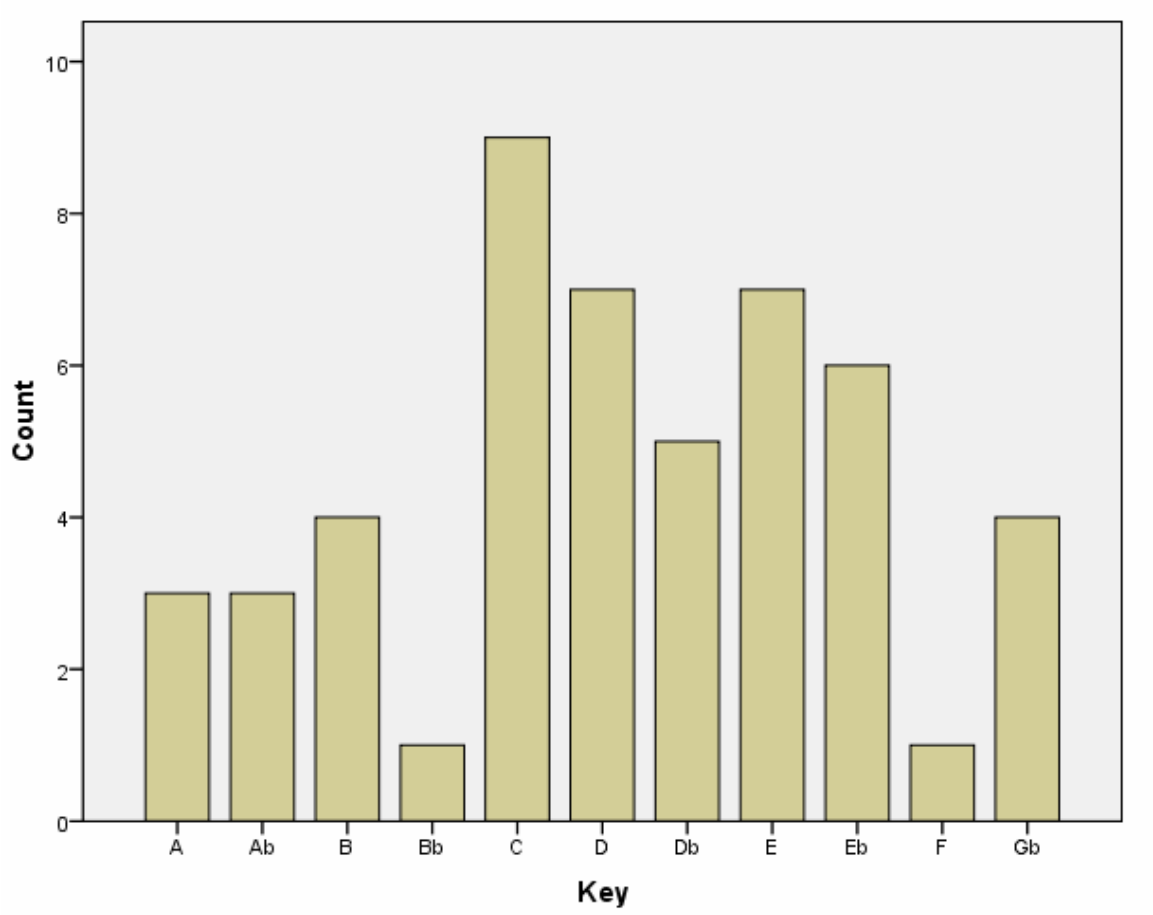


Figure 2.8: Distribution of the Key attribute.

My assumption was that the distribution of the song keys would be relatively even; the fact that they are quite the opposite is indeed a surprise. However, a conscious decision was made not to include this distribution in the programming and to use an even

distribution. The program should be able to produce innovative results, and restricting the key also restricts new sounds (can you really imagine the results being better if the key of G were eliminated?).

The Tempo also produced an interesting result, as it has a bimodal distribution, meaning that it has two distinct ranges in which the values lie (i.e. two modes). See Figure 2.9 showing the distribution.

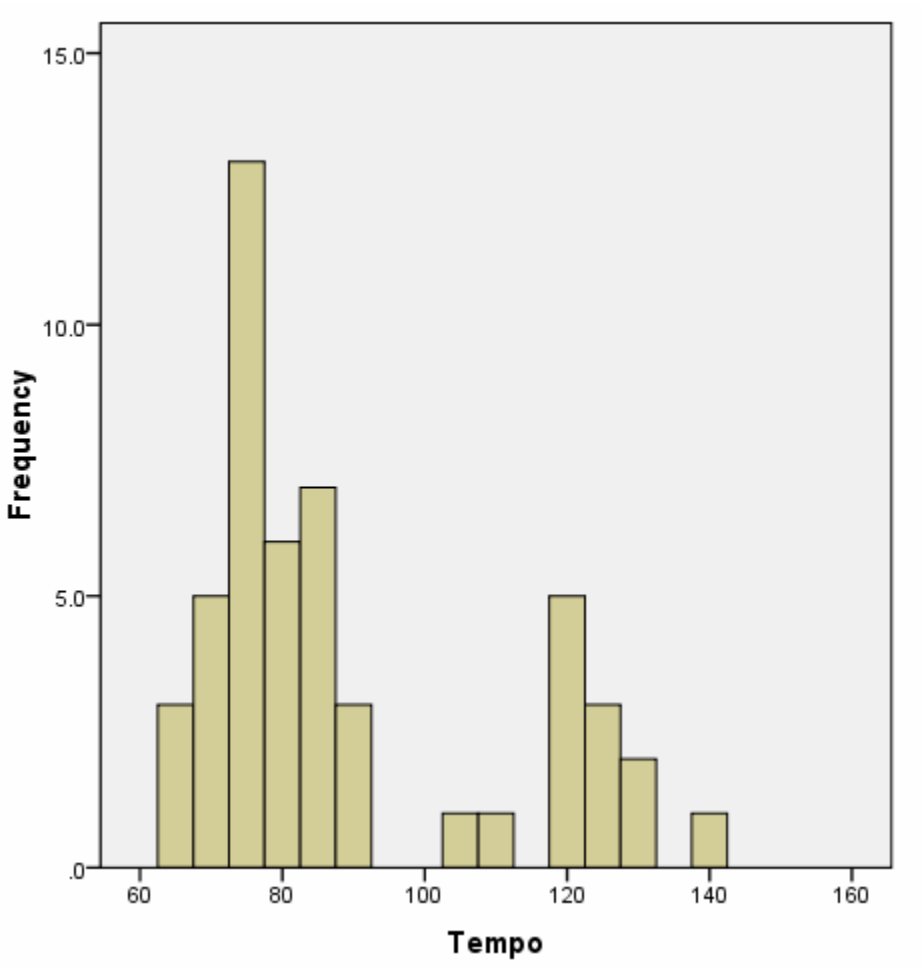


Figure 2.9: Bimodal distribution of the analyzed tempos.

Giving the generated tempos a “wide berth”, an algorithm was used which puts 70% of the tempos between 65 and 90 BPM, 20 % between 120 and 140 BPM, and 10% for the remaining tempos between 90 and 120 BPM.

3 The Program

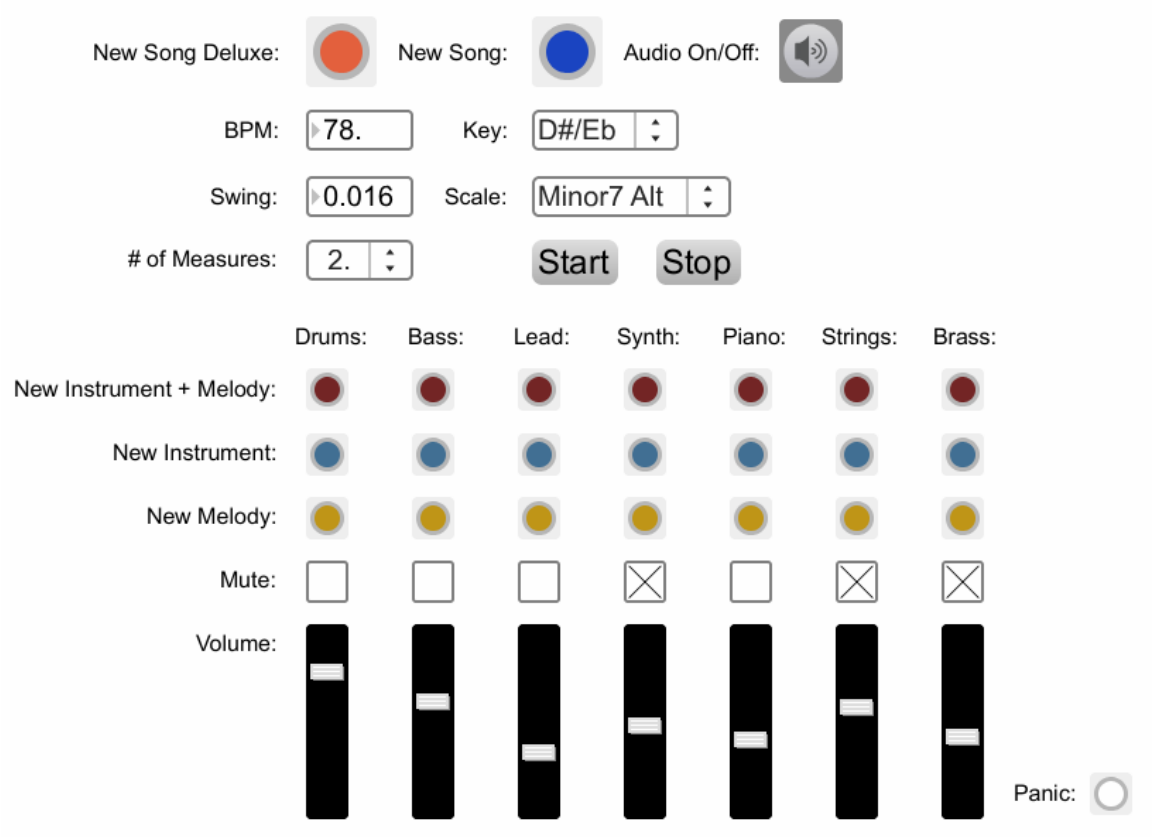


Figure 3.1: User interface view of the Max/MSP program.

3.1 General Overview

The Max/MSP program written for this study takes all of the previously mentioned statistics and incorporates them into a series of algorithms that randomly generate musical phrases. As all of the statistical data was taken from contemporary urban songs, the phrases should also be of the contemporary urban genre. No user input is necessary, as the program already has all composition rules programmed into it; however there are some basic elements that a user can select if he or she wishes, such as key and tempo. The user can also generate an entirely new phrase, parts of the phrase, or

randomly select a new instrument for a phrase. The output of the program includes both generated audio and MIDI files.

The program is a combination of a rule-based program and a data-driven program. A rule-based program is mostly comprised of algorithms that use "if-then-else" clauses; this is when one event occurs (or does not occur) and a subsequent action takes place depending on the outcome of the event (Cope, 2005). This holds true for the program as almost every algorithm is involved in a series of smaller algorithms tied together with if-then-else statements (using the Max/MSP object interface equivalents). Data-driven programming uses an analysis of a database of musical works and then creates new works by using the results of this analysis (Cope, 2005). My program uses an analysis of a musical database as well, but instead of directly using the song analysis data, the data was compiled for a statistical analysis, assigning probabilities to all musical components. Statistical composing is often used to imitate a style or features of a style, and when successful typically creates small segments of music, such as musical phrases or measures, but has the potential for development into a large-scale composition (Cope, 2005). This describes the Max/MSP program, as it generates phrases of contemporary urban music that can be used as a basis for larger compositions, or just as influence for another larger piece of music. See Appendix B for some internal program views.

Max/MSP is a perfect platform for the program. The object-based programming language of Max is ideal for organizing the data flow, as well as for structuring each sub-algorithm within its own unit. It is also a host for VST plugins (used extensively in the program), which allows for a self-contained program without the need to connect it to other external software. As the generated data is almost all related to MIDI parameters,

Max is again superior for working with MIDI and generating both MIDI files and audio output created by the use of those files.

3.2 Musical Elements

Each musical element is calculated in a unique way within the program. Some are based on simple probability while others involve long chains of data before calculating a result. Here is a description of how each of the major program components is calculated in detail.

Tempo, Key, and Swing – All three of these components are calculated in the same way, using a very simple probability algorithm. The key is determined by a random selection of the twelve notes in the chromatic scale of twelve-tone equal temperament, 'A' through 'G#', each receiving an equal chance of being selected. The chosen key is assigned to a value between 0 and 11, which is added to numerical MIDI values in other parts of the program. The swing is a random value between 0 and $1/10^{th}$ of a beat, which is added to every even beat division. This creates a sort of "shuffle" feel to the song, bringing it away from a beat that is exactly on the tempo. The swing parameter uses 100 divisions for that small rhythmic addition, as even slight swing values can significantly change the feel of a song. The tempo is also determined with one simple calculation, but the distribution of the values is not equal. The bimodal distribution of the tempo (as shown in Figure 2.5 and described previously) generates a tempo in beats per minute, which is converted to milliseconds and applied to all pattern timing for the drums and the instruments.

Scale – The scale is randomly selected from 22 different scale types used in the analyzed songs. The basic full forms of the major and minor scales were the most commonly used scales, so they receive the most weight in the random selection (i.e. they are selected more often than the other types). Not all of the 22 scales actually appeared in the songs; a few of them were created as logical additions that were not include in the data set. When a scale is selected, the scale type is sent to other parts of the program that access MIDI note values based on that particular scale.

Drum Combination – Each song from the original database has a particular combination of drum sound types. For example, one song might use a kick, snare and closed hihat, and another might use a kick, clap, open hihat and a shaker. These combinations were taken into account in the program, and the probabilities of each drum type appearing in a song are calculated accordingly. A kick sound is used in all of the songs, so there is a 100% chance that a kick is used in the generated song. Other drum sounds, however, only appear in a subset of the songs, and each has its own probability of appearing in the generated beat. For example, 80% of the original songs include a closed hihat sound, 75% include a clap sound, 50% a snare sound, etc. These are the same chances that they appear in the generated music. As there were no strong correlations among the drum sound types appearing within a song, the program relies on these basic "presence" probabilities for the calculations.

Drum Sound – Once a particular type of drum is determined to be used within a generated song, there are two more tiers of probability calculations before selecting the specific

drum sample to use. First, every drum type has more than one sub-type. A kick drum has four: "punchy" analog, "short" analog, sub, and acoustic. The chances of each of these sub-types being selected is 35%, 25%, 25% and 15%, respectively. Second, once the sub-type has been selected, there is a large database of samples for each sub-type that the program randomly chooses from with an even distribution. Once it is chosen, it is then used as the sample that is triggered for the audio playback of the generated music. This selection process applies to all other drum types, if that drum type was selected to be included in the song.

Instrument Combination – The instrument types were determined in a very similar way to the drums, with different instruments having a specific probability of appearing in the music. A few examples of the instrument types are bass, lead, synth, and piano sounds.

Instrument Sound – The specific sound of the instrument was also determined in a similar way to the drum sounds except that a few of the instruments do not have sub-types. An example of an instrument sub-type distribution is a lead instrument, which has the sub-types analog, "rave", waveform and synth-brass, with the distribution 45%, 35%, 20% and 5%, respectively.

Drum Pattern – The rhythmic patterns of the drums were each considered individually, i.e. the patterns were treated as separate entities for each drum type rather than considering the pattern of all of the drums together in a song as a whole. This allowed for more control over smaller amounts of data, as well as a lot more variety in the

resulting beats. For each drum type, a series of basic patterns were determined by comparing all patterns of that drum together and finding commonalities among those patterns. These basic patterns act as "skeletons" to most of the generated beats, in that they are usually present in their basic form, or as a starting pattern for additional drum-hits to be added. For example, one of the most common basic patterns for an urban kick drum is 1-4-7 (see the Pattern explanation described earlier for clarification). If this pattern is randomly selected from the probability weighted database of kick drum patterns, then there is a 30% chance that it will be this pattern with no additional hits, a 60% chance that it will use this as the base pattern and then add a random number of additional hits to the pattern (between 1 and 5 additional hits), and a 10% chance that it will not use this base pattern at all and instead create a random kick pattern with 1 to 8 hits (with a high chance of the first hit being on the first beat, or "on the one"). The patterns in these three cases could be 1-4-7, 1-3-4-7-9-14, and 1-2-11-15, respectively. Each drum type has its own list of basic patterns and probability distributions for both selecting them and adding additional hits.

Instrument Pattern – The rhythmic patterns of the instruments are determined in a way that is almost identical to the drums, and each instrument had its own set of basic patterns and probability distributions. However, there are some additional considerations for the instruments, which are described below.

Instrument Durations – Based on the type of instrument, the note durations are calculated with one of the following randomly selected types of duration: each note held until the

next note, each note having a length that is a fraction of the time until the next note, a combination of these two, or staccato. The instrument pattern must be determined before the note duration is calculated, with the exception of the staccato notes, which are each a random short length of time in duration. Each instrument type has its own probability of being staccato. For example, a bass is rarely staccato (a 10% chance), while a lead is relatively often staccato (a 40% chance). Drums have no durations, as they are "one shot" triggered samples, playing their full length each time.

Repetition – The repetition of a measure can occur in several different ways:

1) The pattern can be created at the beginning and never change at all

2) The pattern can have a chance to change at the beginning of each measure, but the basic pattern is still the same

3) The pattern can have a chance to change at the beginning of each measure, and an entirely new pattern is created

4) Same as 3) and 4) but the pattern only has a chance to change at certain points and not at the beginning of each measure

The repetition schemes for the drums and the instruments are the same, except that an instrument has the following additional possibilities:

5) Same as 1) and the notes also never change

6) Same as 1) but the notes change

7) If the pattern does not change in the instances of 2), 3) and 4), the notes still have a chance to change.

Polyphony – The polyphony of an instrument can be in solo mode (one voice at a time), chord mode (playing more than one voice at a time), or a combination of both (blending chords with solo notes in a phrase). The default is solo mode, and the chances of being in chord mode are different for each instrument. For example, a bass is never in chord mode, and a piano is in chord mode 70% of the time. If the instrument is in chord mode, then up to three notes within the chosen scale are played together, having the same duration. Polyphony is only relevant to the instruments, as the individual drums are always in solo mode.

Dynamics – The volumes of the different instruments can be changed by the user. Each has a default volume that is loaded with the program, which is about an average of the typical volume used for that sound. The velocities of the MIDI notes are created by generating a random velocity value within a small range of numbers, and that range being within the one appropriate for that sound type. The default volumes of the sounds are usually sufficient to get a general idea of the song, but it is often necessary to adjust the volumes to fine tune the overall mix.

Note Melodies – Note melodies are determined by a combination of the scale used and the average deviation of the following notes (the average note intervals between each successive jump in the melody). Depending on the scale, the weighted distribution of the possible interval from one note to the next changes, so that some melodies do not stray very far while others climb up and down almost two octaves. The most common average interval jump in the analyzed songs was a 3rd, and this in turn is the most common

average interval jump for most instruments and scale types in the program. The initial notes have a chance of being the tonic note (or root note) of the selected scale, and the chances of that occurring depend on the particular instrument.

4 Results

4.1 Does It Work?

Starting with a purely subjective view, the answer is 'yes'. Each generated musical phrase sounds like contemporary urban music, using the right sounds, playing genuine beats, and creating melodies that fit well into the genre. This is not to say, however, that every generated song is good music. In fact, only about one in five generated songs has something good or interesting enough to be developed further, and only about one in twenty is good enough to use as a relatively solid basis for a song. The other phrases it generates have the right musical elements, but the music itself does sound quite "random". While this may sound inadequate, this should be considered as a good rate. The time it takes to generate a song with the program versus the time it would take to come up with an idea and create it from scratch is astounding. One should also keep in mind that a composer might also come up with several ideas before deciding one is good enough to develop further.

Two additional observations were made. First, when listening to the generated music, the high quality of the sounds used in the program makes up for some of the possible "randomness" in the result. When the sounds have a pleasingly good/genuine quality, the overall impression of the music becomes more favorable. Second, when composing urban music, it is common to use an external interface such as an MPC unit or a padded MIDI input device like Native Instruments' "Maschine" hardware. This automatically limits the way that the music can be played and recorded, such as a limited range of notes or a limited number of drum samples. The Max/MSP program limits the music in similar ways, thus sharing some of the same compositional characteristics.

However, an objective viewpoint is the one that can legitimately show that the results are urban. Here is an example of the evaluation of a randomly generated 2-bar musical phrase, examining the properties of each of the generated parts and comparing them to the original data collected from the top 50 contemporary urban songs.

Figure 4.1 shows the generated kick drum part with musical notation.

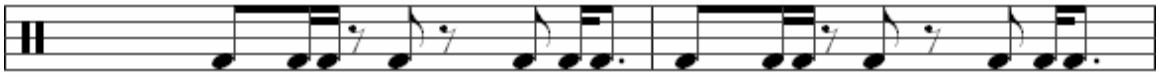


Figure 4.1: Kick drum notation for the 2-bar phrase example.

The numerical string representation of this kick pattern is 1-3-4-7-11-13-14 for both measures. This exact pattern never appeared in a song in the original data, but the basic pattern of 1-4-7-11-14 does appear often within the kick patterns of the songs (as well as several subsets). Assuming that this pattern or one similar was used as the backbone of the kick pattern, the additional hits were randomly generated around that pattern and created the new pattern here.

Figure 4.2 displays the remaining elements of the beat.

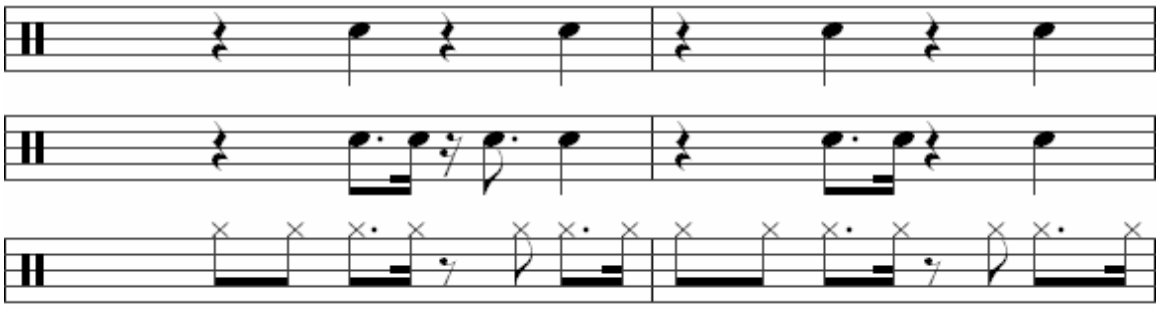


Figure 4.2: Phrase example notation for the Clap, Snare and Closed Hihat, respectively.

The clap pattern 5-13 is by far the most common one used in the songs, and it is reflected in this notation (which translates to a clap sound on the 'two' and the 'four' of a measure). The snare pattern here is 5-8-10-13 in the first measure and 5-8-13 in the second measure. Both match snare drum patterns that have been used in the analyzed songs. The closed hihat pattern is 1-3-5-8-11-13-16. This does not match any pattern in the songs, nor does it use one of the basic patterns. The closed hihat has a higher chance than most other drum of a random pattern generation without any basic pattern, so this must be the case here. Many of the songs have closed hihat patterns that are similar, in that they do not follow an obvious pattern yet they have several hits per measure.

The first melodic element shown here in Figure 4.3 is the bass pattern.

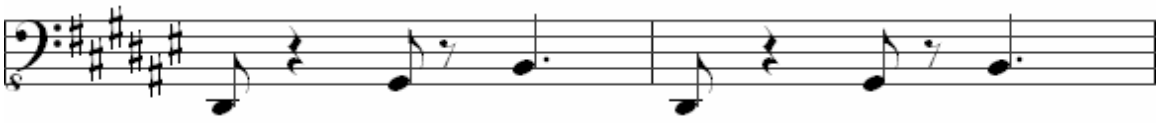


Figure 4.3: Bass notation for the phrase example.

The song is in the key of D# Minor (or F# Major), and the key signature reflects this. The rhythmic pattern is 1-7-11 for both measures, which is a common pattern used in the songs as well as one of the basic patterns in the collection. Assuming the key is D# Minor, the bass starts on the root note D#, which is the case in approximately 50% of the analyzed songs. The interval jumps a fourth and then a third before repeating the measure, which also fits well. The note durations are a combination of notes that hold until the next note is played and notes that play for a shorter time (an option mentioned in

the note duration explanation). This is not as common in the songs, but does indeed occur.

Here in Figure 4.4 is the notation for the lead pattern.

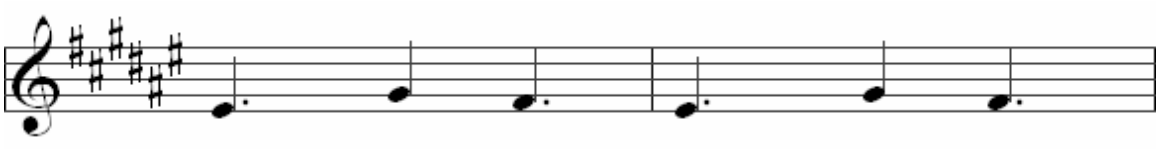


Figure 4.4: Lead notation for the phrase example.

It has a similar evaluation as the bass above, with the rhythmic pattern being a common one (1-7-11) and coincidentally "shadows" the bass line in this musical phrase. In this case, the notes are all held until the next ones are played, i.e. there are no rests in the pattern.

The synth pattern looks like the notation in Figure 4.5.

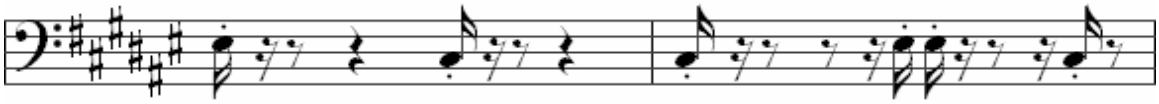


Figure 4.5: Synth notation for the phrase example.

Here you will notice that the notes are staccato. The synth sound in the song collection is staccato over 50% of the time. Both the pattern and the notes are different in each measure. The 1-9 pattern of the first measure is found in the basic patterns for the synth instrument, but the second pattern 1-8-9-14 is most likely the same basic pattern of 1-9 with two additional random hits included.

Finally, the piano pattern is shown here in Figure 4.6.

Figure 4.6: Piano notation for the phrase example.

This is the only pattern that uses chords in this musical phrase, and the piano uses chords often in the analyzed songs. The pattern is 1-5, which either is a build from the basic pattern of simply 1 (heard often in piano chord lines) or was generated randomly without a basic pattern. The notes in the chords follow the D# Minor scale, and the spacing between the notes is slightly closer than usual, but not uncommon.

The musical properties of the generated phrase match the properties of the original data set, showing how the algorithms in the program calculated the music to match the genre "correctly". As the sounds included in the program are all typical urban sounds, the fact that the sounds are appropriately chosen for the musical phrase is no surprise, nor is it that they are similar to the ones evaluated. The drum & instrument combination and the sound types are actually quite similar to one particular song from the database, although the patterns and melodies used are very different.

4.2 Conclusion

When the rules for a genre are properly defined and implemented by using stochastic algorithms derived from the analysis of a collection of songs, a computer program can be created to compose music for that genre using those rules. It is more

feasible to define the rules for contemporary urban music than some other genres because the composition style uses repetition and the types of sounds used are somewhat restricted, unlike genres such as jazz or progressive rock. Urban music is also more adaptable to this kind of program because the sounds are typically electronic, or at least the composition and production are typically done with a computer. This is only an argument for having a similar sound, however, as the ideas for acoustic based genres could still be generated with a computer but performed using acoustic instruments.

4.3 Future Work

As the program is now, it creates musical phrases but not full songs with musical structure. An additional set of rules could be implemented into the program such that full songs could be created after a structural analysis of the original songs is performed. Shorter musical phrases are easier and faster to analyze, making them ideal for a study of this nature.

The melodic analysis of the music consisted of starting and ending notes of phrases, and average interval jumps. While this proved to be satisfactory, a third or fourth order Markov chain would possibly reduce the number of results that have unconvincing notes in the melodies. However, using only a Markov chain would also reduce the chances that program generates an innovative melody, as the Markov algorithm would ensure that a similar melodic line had already existed. The solution would be to change the program so that both options are available, the current "average interval jump" method and a Markov chain method, where each has a chance of being used for either single instruments or the entire musical phrase.

To increase the ratio of favorable to unfavorable output, a simple genetic algorithm function could be added where a user claims whether a phrase sounds good or bad, or some other kind of ranking system. The program would record the parameters chosen for that phrase and eventually a database would form, which could be analyzed to improve the program by changing the algorithms, adjusting the probability functions, or changing the selection of available sounds.

The program relies on the capabilities and sounds of commercial VST plugins. A function of the program could be to generate sounds from scratch, without the need to access additional plugins, and then the program would be able to create everything on its own and be self-contained. However, this extremely large task would require much development time.

Finally, the contemporary urban genre was the only one used for this Max/MSP program. An obvious next step would be to create algorithms that generate music for other genres.

5 Bibliography

Abletat. "Abletat Muse." 2009 <http://www.abaltat.com/home.php?cat=255>

Billboard. March 2009 <http://www.billboard.com/bbcom/index.jsp>

Bohn, James. "ILLIAC I." 2009 <http://ems.music.uiuc.edu/history/illiac.html>

Contactmusic.com. "Chris Brown – RnB Star Chris Brown Named Artist Of The Year." 2008 <http://www.contactmusic.com/news.nsf/article/r n b star chris brown named artist of the year_1089351>

Cope, David. "Computer Modeling of Musical Intelligence in EMI." Computer Music Journal Vol. 16, No. 2, Summer 1992: 69-83

Cope, David. Computer Models of Musical Creativity. Cambridge, MA: The MIT Press, 2005.

Cope, David. Virtual Music. Cambridge, MA: The MIT Press, 2001.

Cycling '74. 2009 <http://www.cycling74.com>.

David Cope. UCSC. 2009 <http://arts.ucsc.edu/faculty/cope/>.

iTunes Store. March 2009 <http://www.apple.com/itunes/>.

Kramarz, Volkmar. The Pop Formulas. Bonn, Germany: Voggenreiter, 2007.

Microsoft Research. "Songsmith." 2009 <http://research.microsoft.com/en-us/um/redmond/projects/songsmith/>

Native Instruments. 2009 <http://www.native-instruments.de>.

Nierhaus, Gerhard. Algorithmic Composition: Paradigms of Automated Music Generation. Vienna, Austria: Springer, 2009.

PG Music Inc. "Band-In-A-Box." 2009 <http://www.pgmusic.com/bandbox.htm>

Rayzoon Technologies LLC. "Jamstix 2" 2009 <http://www.rayzoon.com/jamstix2.html>

Rowe, Robert. Machine Musicianship. Cambridge, MA: The MIT Press, 2001.

Sadun, Erica. Ars Technica. "Electric sidemen: a look at Microsoft Songsmith." 2009 <http://arstechnica.com/microsoft/news/2009/01/microsoft-songsmith-review.ars>

Sloboda, John A. Generative Processes in Music The Psychology of Performance, Improvisation, and Composition. New York: Oxford UP, USA, 2001.

SoundTrek. 2009 <http://www.soundtrek.com/content/index.php>

Taube, Heinrich. Notes from the Metalevel An Introduction to Computer Composition (Studies on New Music Research). New York: Routledge, 2004.

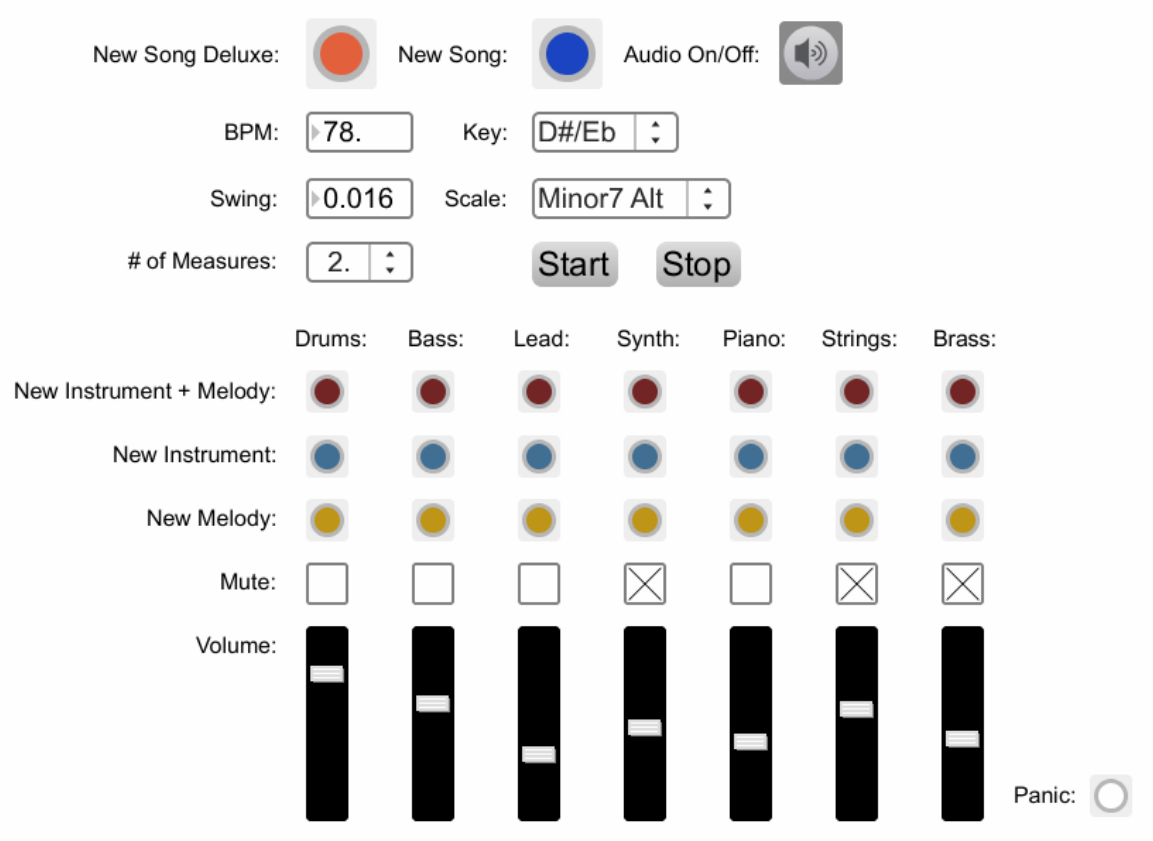
Wikipedia. 2009 <http://www.wikipedia.org/>.

6 Appendices

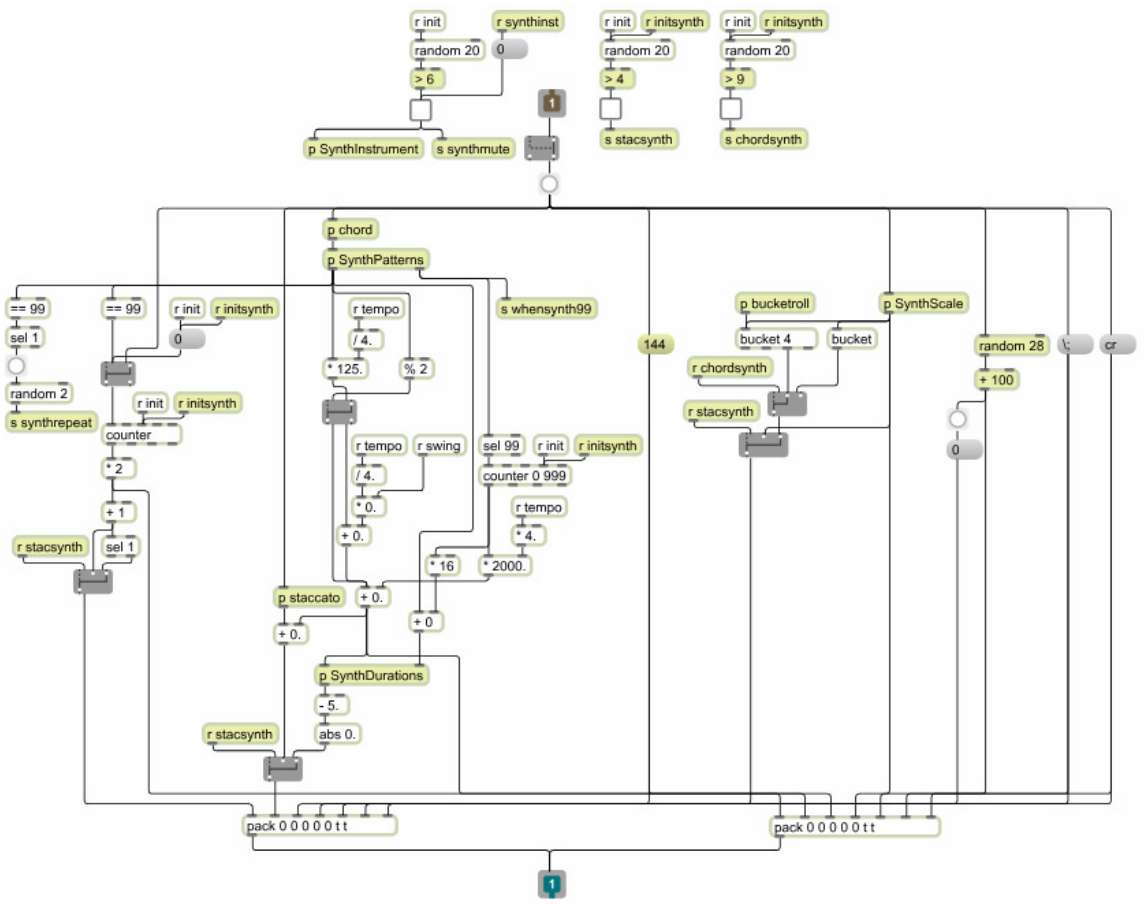
Appendix A – List of Analyzed Songs

Rank:	Chart:	Artist:	Song:	Album:
1	Billboard Hip Hop / RnB	Jamie Foxx	Blame It	Intuition
1	Billboard Rap	T.I.	Dead and Gone	Paper Trail
2	Billboard Hip Hop / RnB	Keri Hilson	Turnin Me On	Single
2	Billboard Rap	Soulja Boy Tell 'em	Kiss Me Thru The Phone	iSouljaBoyTellem
2	iTunes RnB/Soul	Akon	Beautiful	Freedom
3	Billboard Hip Hop / RnB	The-Dream	Rockin' That Thang	Love Vs Money
3	Billboard Rap	Kanye West	Heartless	808s and Heartbreak
3	iTunes Hip Hop / Rap	Asher Roth	I Love College	Single
4	Billboard Hip Hop / RnB	Beyonce	Diva	I Am...Sasha Fierce
5	Billboard Hip Hop / RnB	Ne-Yo	Mad	Year of the Gentleman
5	Billboard Rap	Yung L.A.	Ain't I	Single
5	iTunes RnB/Soul	The-Dream	Walkin' On the Moon	Love Vs Money
6	Billboard Hip Hop / RnB	Bobby V.	Beep	The Rebirth
6	Billboard Rap	Eminem	Crack A Bottle	Single
6	iTunes RnB/Soul	Akon	I'm So Paid	Freedom
7	Billboard Rap	T.I.	Live Your Life	Paper Trail
7	iTunes Hip Hop / Rap	Kid Cudi	Day 'n' Nite	Kid Cudi
7	iTunes RnB/Soul	Akon	Right Now (Na Na Na)	Freedom
8	iTunes Hip Hop / Rap	Maino	All the Above	Single
9	Billboard Rap	GS Boyz	Stanky Legg	Single
10	Billboard Rap	T.I.	Whatever You Like	Paper Trail
10	iTunes Hip Hop / Rap	Kanye West	Love Lockdown	808s and Heartbreak
10	iTunes RnB/Soul	Ne-Yo	Closer	Year of the Gentleman
11	Billboard Hip Hop / RnB	Jamie Foxx	She Got Her Own	Single
13	iTunes RnB/Soul	Estelle	American Boy	Shine
14	Billboard Hip Hop / RnB	Jamie Foxx	Just Like Me	Intuition
15	Billboard Hip Hop / RnB	Musiq Soulchild	So Beautiful	On My Radio
16	Billboard Hip Hop / RnB	Jennifer Hudson	If This Isn't Love	Jennifer Hudson
17	iTunes Hip Hop / Rap	T-Pain	Freeze	Single
19	iTunes Hip Hop / Rap	Flo Rida	Low	Mail On Sunday
19	iTunes RnB/Soul	Chris Brown	Forever	Single
20	Billboard Hip Hop / RnB	Anthony Hamilton	Cool	The Point Of It All
22	iTunes Hip Hop / Rap	Chamillionaire	Creepin'	Single
23	Billboard Hip Hop / RnB	Keyshia Cole	Playa Cardz Right	A Different Me
23	iTunes Hip Hop / Rap	Kanye West	Stronger	Graduation
24	Billboard Hip Hop / RnB	Ne-Yo	Miss Independent	Year of the Gentleman
25	Billboard Hip Hop / RnB	Jennifer Hudson	Spotlight	Jennifer Hudson
26	Billboard Hip Hop / RnB	Plies	Want It, Need It	Da REAList
26	iTunes RnB/Soul	The-Dream	Let Me See the Booty	Love Vs Money
27	Billboard Hip Hop / RnB	OJ Da Juiceman	Make Tha Trap Say Aye	The Otha Side of the Trap
27	iTunes Hip Hop / Rap	Mike Jones	Next to You	Single
28	iTunes Hip Hop / Rap	Young Jeezy	My President	The Recession
29	iTunes Hip Hop / Rap	T.I.	Swagga Like Us	Paper Trail
30	iTunes Hip Hop / Rap	Savage	Swing	Savage Island
31	iTunes Hip Hop / Rap	Flo Rida	In The Ayer	Mail On Sunday
33	iTunes Hip Hop / Rap	Pitbull	Krazy	Single
36	iTunes Hip Hop / Rap	Soulja Boy Tell 'em	Crank That	iSouljaBoyTellem
38	iTunes Hip Hop / Rap	Lil Wayne	Got Money	Tha Carter III
41	iTunes Hip Hop / Rap	Lil Wayne	Lollipop	Tha Carter III
42	iTunes Hip Hop / Rap	Lil Wayne	A Milli	Tha Carter III

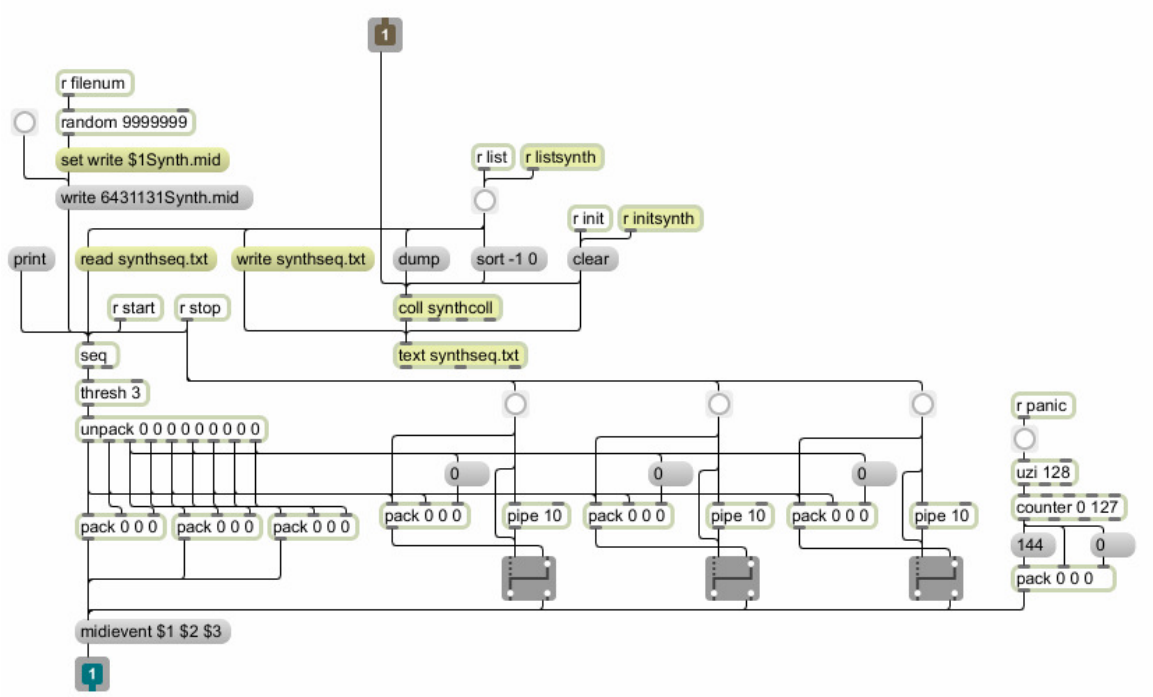
Appendix B – Max/MSP Program Views



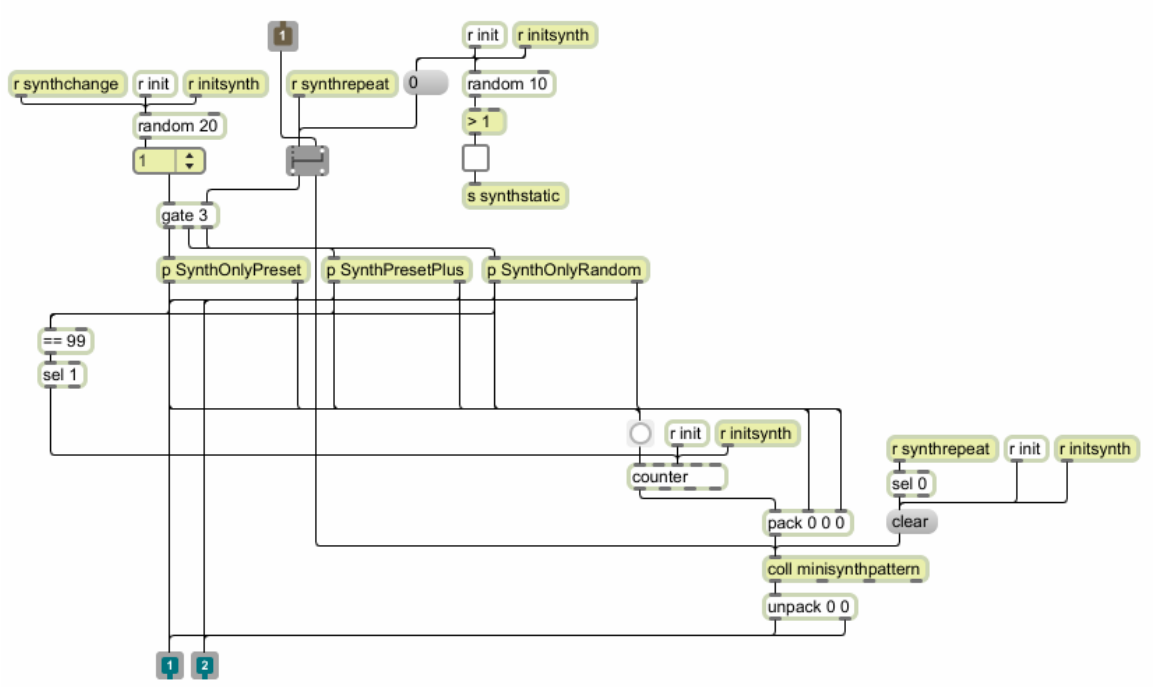
Appendix B 1: User interface view of the Max/MSP program.



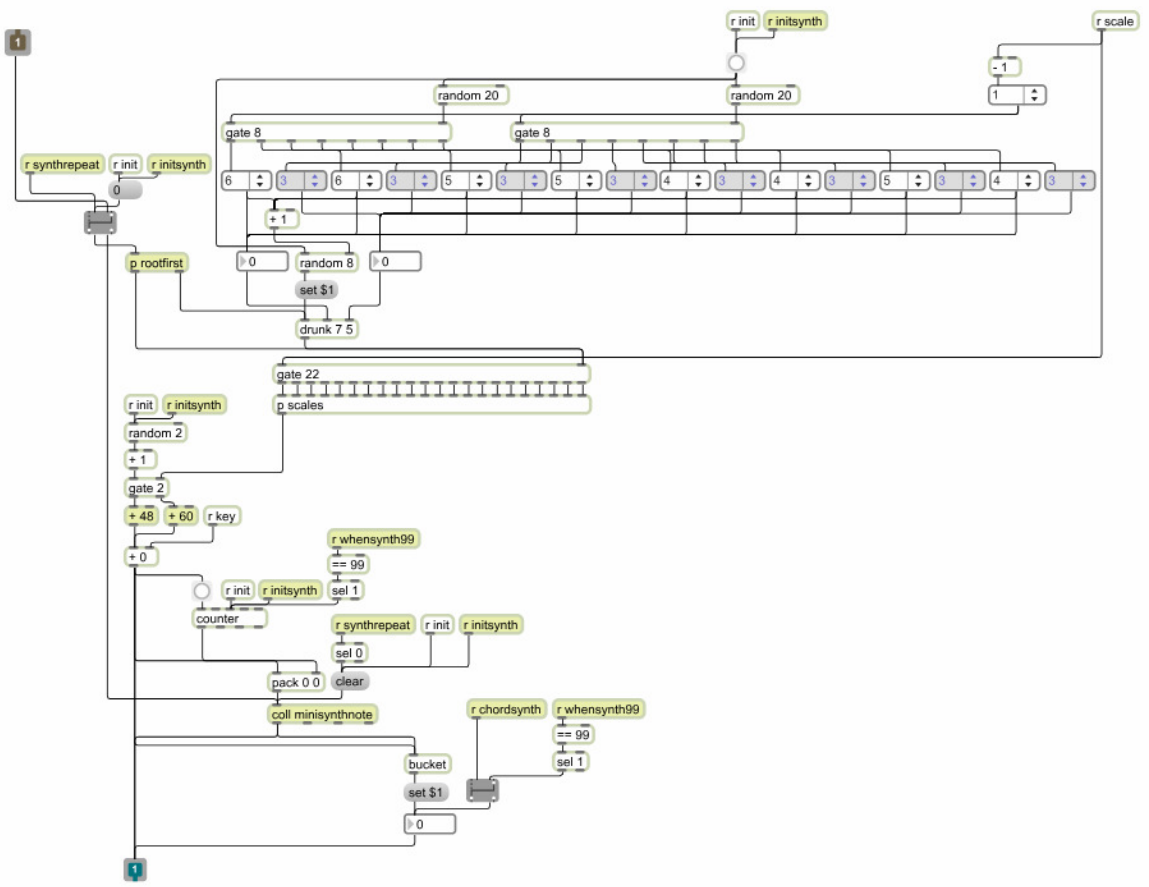
Appendix B 2: Internal Max/MSP layout of the Synth generating patch.



Appendix B 3: Internal Max/MSP layout of the Synth MIDI file creating patch.



Appendix B 4: Internal Max/MSP layout of the Synth pattern patch.



Appendix B 5: Internal Max/MSP layout of the Synth melody patch.