Interactive Music: Compositional Techniques for Communicating Different Emotional Qualities

Robert Winter
James College

University of York, UK
June 2005

4th Year Project Report for degree of MEng
in Electronic Engineering
with Music Technology Systems

Abstract

Music plays a large part in how a dramatic medium is perceived and has a huge potential to heighten emotion. Composers attempt to communicate emotions to a listener through the use of structural compositional techniques such as mode, rhythm, instrumentation and harmony. This report introduces and discusses these techniques. It goes on to describe a software program, developed in the Pure Data programming environment, which influences the emotional expression of a musical score in real-time. There are proposed future applications such as in the computer gaming industry, where real-time control over the structural makeup of the soundtrack could provide new levels of immersive game-play.

Sammanfattning

Musik spelar en stor roll för hur olika typer av multimedia upplevs och har en enorm potential att förhöja känslan. Kompositörer försöker förmedla känslor till lyssnaren genom att använda sig av kompositionstekniker så som mode, rytm, instrument och harmonier. Denna rapport introducerar och diskuterar dessa tekniker. Vidare beskriver den ett dataprogram, utvecklat i utvecklingsmiljön Pure Data, som påverkar ett musikstyckes emotionella uttryck i realtid. Framtida applikationer föreslås så som dataspelsindustrin där realtidskontroll över musikstyckets utseende kan förhöja spelkänslan.

Table of Contents

1 Introduction

The field of interactive music for multimedia applications moves away from the trend of detached, linear scores usually found in film soundtracks, towards a point where a musical score can adapt in real-time to the actions of external factors. The computer game industry is an important field in interactive multimedia, and strives to create true immersive game-play for a user by creating beautiful visual environments and lifelike scenarios. Audio soundtracks in their simplest form are usually based around a simple audio engine that allows loops of pre-composed musical phrases to be called upon to play using nametags to represent the current state of play. When the state of play changes, the current audio loop is either truncated with a specific pre-composed ending or cross-faded with the next audio loop if required. Enhancements to this simple audio engine model include the technique of horizontal re-sequencing (O'Donnell, 2002) where pre-composed sections of music are repositioned in time according to the state of game-play. Another technique used is vertical re-orchestration that involves the dynamic mixing of the instrumentation of a pre-composed audio loop to reflect the decisions made by the user.

The common techniques described above are all based around pre-composed sections of music that are shifted in time or remixed to reflect an ever-changing storyboard. Investigations into improving user interactivity beyond its current state, through the use of an interactive audio soundtrack, have been limited.

The MEng thesis, Interactive Music for Computer Games (Hobbis, 2003) provides a start into investigation of a musical soundtrack for a computer game that adapts according to the current state of game play. Adjectives such as 'danger' and 'tense' are used to describe situations that a user could find themselves in. These adjectives are also used to define different areas of a user interface that controls the basic structural changes applied to a melody line.

This project investigates how the structure of a musical score could be altered further to enhance the emotional feedback given to a user to improve the immersion in a multimedia environment. The specific nature of changes to the musical structure of a pre-composed musical score can be investigated using established and current research in the field of music and emotion.

1.1 Music and Emotion

Music plays a large part in how a dramatic medium such as a multimedia application is perceived and has a huge potential to heighten emotion. Much attention has been paid by researchers as to the expressive qualities of music, specifically focusing on the expression of emotion. A definition of emotion is given by Juslin (2004): "..emotions can be seen as relatively brief and intense reactions to goal-relevant changes in the environment..."
Two areas of interest can be found from this research. Firstly, what emotions can be consistently expressed in music by means of a performance and secondly, what are the structural factors in music that contribute to the perceived emotional expression? Tools

used by a performer to express emotion include changes in mean tempo, timing and sound level. Structural factors include mode, harmony and rhythm.

Composers write music to try to achieve different intended emotions to be communicated to a listener through the use of compositional techniques. Listeners almost always judge the perceived emotional expression of a composition by means of a musical performance. This suggests that the perceived emotional expression is dependent on both the properties of the composed structure and the emotional colouring introduced by the performer. This raises the question: How do listeners react to music? To begin to understand this, a distinction between the listeners' perception of emotional expression and a listeners' emotional reaction (induction) needs to be made. Perception refers to a listener recognising say, a sad expression in music. Induction refers to a listener feeling sad when hearing music. Juslin (2004) states that the types of emotion typically expressed and perceived in music may be different from those typically induced by music.

1.2 The Current Situation

Much research has been carried out concerning the perceived emotional expression from the *performance* of music. Models have been created and recently implemented to allow real-time control of the aspects of human performance to produce a computer controlled synthesis of performance.

Many of the expressive qualities of music are defined by the structural changes that occur in a written score. Few applications exist that allow real-time control over some of the structural factors of a composition that a composer uses to communicate different perceived emotional expressions. The possibility that transitions from one perceived expression to another can be achieved dynamically across time using real-time structural changes is an interesting proposition. No systems implementing the findings of current research in the area of the effect of musical structure on perceived emotional expression exist.

1.3 Justification and Motivation

Scope exists to create a system that gives real-time emotional feedback to a user using compositional tools highlighted in existing research such as rhythmic, timbral and harmony variations.

It can be shown that music can influence people's action tendencies (Juslin, 2004). It follows, therefore, that if a piece of music can be altered to communicate different moods or emotions, the method can then be applied to many scenarios such as game soundtracks to represent the current state of play. Likewise, interactive musical installations could use a system where the amount of movement of a person is sensed and the mood of the output sound reflects this. An excited, physically active person would steer the desired musical output to express a more bright, happy and exhilarated performance where the opposite effect could be a calm and dreamy performance.

Specifically, this project looks into the possibility of investigating the structural compositional changes that could be applied to a pre-composed piece (e.g. rhythmic

alterations, mode, melodic progression etc) to communicate different perceived emotional expressions. What effect does the interaction of certain factors have upon the perceived emotional expression? This question could be investigated using a system that allows systematic manipulation of structural changes to be applied in real-time to a pre-composed piece of music in a musical context. This system aims to add to research and development in the area of musical structure and automatic synthesis of emotional expression.

1.4 Report Structure

Section 1 aims to provide an introduction to the work carried out during the project.

Section 2 provides a literature review focusing on the field of music and emotion.

Section 3 outlines the project aims and objectives.

Section 4 presents work carried out investigating how structural changes in a musical score can be implemented to communicate different emotional qualities.

Section 5 presents a discussion regarding suitable development environments.

Section 6 describes how data from an input score can be used within `pd`.

Section 7 describes how accompaniment and bass parts are generated in real-time to fit the score chord progression.

Section 8 explains how the rhythm and percussion part of the score is changed according to user input.

Section 9 describes the implementation of a simple algorithm to change the mode of the input score.

Section 10 presents an implementation to change the instrumentation of the melody line according to user input position.

Section 11 describes how user input is used to control all of the structural changes being applied to the input score by means of interpolation in an Activity-Valence control space.

Section 12 gives detail on how the system developed in this project is integrated with pDM, an existing program that gives real-time control over the performance of a fixed musical score.

Section 13 presents system testing and user acceptance testing.

Section 14 presents the conclusions of the project.

Section 15 outlines further work that could be carried out to further development.

A listing of references and bibliography, along with appendices are found at the end of the report.

2 Literature Review

This section presents a summary of literature in the field of music and emotion, specifically focusing on reviews of emotional expression in musical performance and composition.

2.1 Communication of emotions in music performance

A review of research on communication of emotions in music performance by Juslin (2001) indicates that the same musical score can be performed in many ways to alter the perceived emotion by the listener. Cues such as mean tempo, articulation, small timing variations and vibrato are utilised by performers in the communication of emotion. These cues are the tools that performers can use to interpret a musical score into a performance to convey their intentions. Listeners then reinterpret a performance to form his or her own ideas. This process is illustrated in figure 1.

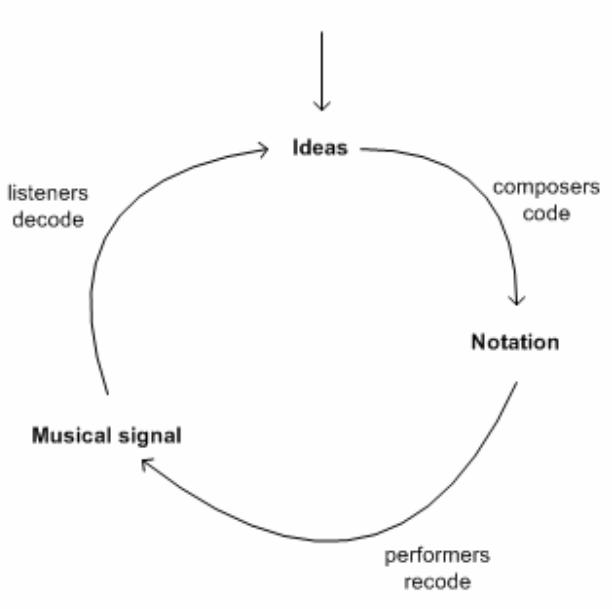


Figure 1: Cycle of communication in music.

A composer can use cues such as mode, rhythm and harmony to encode their ideas into a musical score for interpretation by a performer.

A theoretical framework is presented by Juslin based upon the work on non-verbal communication of emotions by Egon Brunswick. An adapted version of the 'Lens Model' shown in figure 2, describes the communicative process used by performers and listeners to judge emotional expression. The performer has certain expressive intentions that are encoded into a performance that contains expressive cues. The listener then decodes these cues to form a judgement of emotional expression.

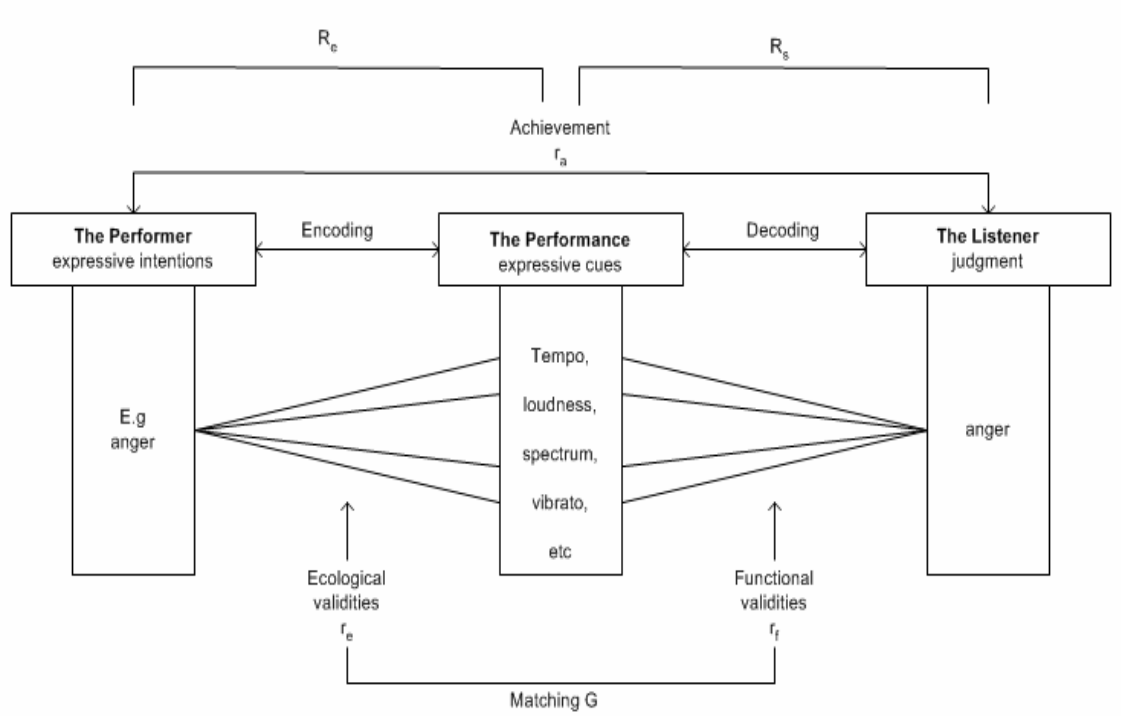


Figure 2: The Brunswikian lens model for communication of emotion in music (Juslin, 2001).

The expressive cues are described as being probabilistic. This means that they are not very reliable in communicating emotion. For a reliable listener judgement to be formed, the cues have to be combined. It is noted that listeners use different cues at different times, possibly depending upon which cues are communicated the strongest. Therefore, Juslin suggests that performance cues contribute independently to the perceived emotion of the listener, and that there is little interaction between cues. This proposal is in contrast to that of Gabrielsson & Lindström (2001) who suggest that when altering the *melodic* properties of a score, emotional expression is almost always dependent on a combination of compositional cues. These findings regarding musical structure are presented at a later point in this report.

The measure r_e represents the *ecological validity* of the cue, and measures the validity of the cue in predicting the performers' intention. The measure r_f represents the functional validity of the same cue, and measures the validity in predicting the listeners' emotion judgment. Achievement r_a is the measure of the relationship between the performers' intention and the listeners' judgement, and can be used to evaluate the accuracy of emotion communication. A factor G is used to compare the similarity between the ecological and functional validities – are the performer and the listener using the same code (i.e. on the same 'wavelength')? These index values are all used in the Lens Model Equation to give an indication to how successful the communication of intended emotion to a listener is. Questions can then be asked if the intended emotion is not being communicated correctly, as to whether the performers are using a different code to that of

the listener; the performer is utilising the code inconsistently; or the listener is applying the code inconsistently.

2.2 Communication of emotion through musical structure

A review of the influences of musical structure on perceived emotional expression (Gabrielsson & Lindström, 2001) presents investigations carried out by Hevner. This work resulted in the well-known 'adjective cluster' (Figure 3), comprising of many adjectives describing emotional states in eight clusters in a circular arrangement. Adjacent clusters are designed to be similar and a progression though the numerical clusters to an opposite state is supposed to represent a contrasting emotion.

Figure 3: Hevner's Adjective Cluster (Adapted from Gabrielsson & Lindström, 2001)

Hevner conducted experiments playing short tonal pieces of music to hundreds of subjects. Each heard the original plus manipulated versions differing in either (1) *mode*, major and minor; (2) *melodic direction*, ascending vs. descending; (3) *harmony*, simple consonant harmony vs. complex dissonant harmony; (4) *rhythm*, firm (chords on the beat) vs. flowing (broken chords); (5) *tempo*, fast vs. slow; and (6) *pitch level*, one octave or more apart.

Listeners were asked to mark the adjectives that described the piece played. Differences in choice of adjectives between any pair of contrasting pieces could then be therefore due to the manipulated variable. Hevner's results from the studies are summarised in Table 1.

Musical Factor	Dignified/ Solemn	Sad/ Heavy	Dreamy/ Sentimental	Serene/ Gentle	Graceful/ Sparkling	Happy/ Bright	Exciting/ Elated	Vigorous/ majestic
Mode	Minor 4	Minor 20	Minor 12	Major 3	Major 21	Major 24	-	-
Tempo	Slow 14	Slow 12	Slow 16	Slow 20	Fast 6	Fast 20	Fast 21	Fast 6
Pitch	Low 10	Low 19	High 6	High 8	High 16	High 6	Low 9	Low 13
Rhythm	Firm 18	Firm 3	Flowing 9	Flowing 2	Flowing 8	Flowing 10	Firm 2	Firm 10
Harmony	Simple 3	Complex 7	Simple 4	Simple 10	Simple 12	Simple 16	Complex 14	Complex 8
Melody	Ascend 4	-	-	Ascend 3	Descend 3	-	Descend 7	Descend 8

Table 1: Summary of Hevner's results from six experiments. (Adapted from Gabrielsson & Lindström, 2001)

Further experiments carried out by many others (see Gabrielsson & Lindström, 2001) investigated the effect of structural changes in music composition on perceived emotion. The results are summarised below in section 2.3.

2.3 Summary of perceived emotional expression of compositional transformations

Each structural compositional change investigated is listed below, along with the perceived emotional expressions from that change. Section 2.9 presents a model that summarises the main structural changes and the perceived emotional expression from each change in the form of an Activity-Valence control space.

2.3.1.1 Mode

Happiness and joy may be associated with major mode whereas sadness with a minor mode from the ages of around 7 years old. Major mode can also be associated with emotions such as graceful, serene and solemnity and minor mode with dreamy, dignified, tension, disgust and anger.

2.3.1.2 Pitch

A high pitch register may be associated with emotional expressions such as happy, graceful, serene, dreamy and exciting, and also with surprise, potency, anger, fear and activity. A low pitch register can be associated with sadness, dignity, solemnity, vigour and excitement, and further boredom and pleasantness. Large pitch variations may be associated with happiness, pleasantness, activity or surprise while small variations in pitch can be associated with disgust, anger, fear or boredom.

2.3.1.3 Melody

A wide melodic range in a piece may be said to express joy, whimsicality and uneasiness whereas a narrow melodic range may be associated with expressions such as sad, dignified, sentimental, tranquil, delicate and triumphant.

With regard to melodic direction or melodic contour, an ascending melody line may be considered to relate to dignity, serenity, tension and happiness along with fear, surprise, anger and potency. A descending melody may be associated with excitement, gracefulness, sadness and vigour. In the experiments carried out it was suggested that melodic direction has little if any effect on emotional expression. Little research has been carried out in the area regarding melodic motion. Juslin suggests that stepwise melodic motion may be associated with dullness and melodic leaps may suggest excitement. Lindström (1997) found an interaction between melodic contour and rhythm affect listener judgements of happiness and sadness.

2.3.1.4 Harmony

Simple and consonant harmonies suggest expressions such as happy, relaxed, graceful, serene, dreamy, dignified and majestic. Complex and dissonant harmonies may be associated with excitement, tension, vigour, anger, sadness and unpleasantness.

2.3.1.5 Tonality

Chromatic harmonies suggest sad and angry emotions. Tonal melodies could be perceived as joyful, dull and peaceful whereas atonal could be angry.

2.3.1.6 Intervals

For harmonic (simultaneous) intervals with respect to consonance and dissonance the perceived emotions correspond to those found under harmony in section 2.3(d) For results concerning pitch level of the interval (high/low) see section 2.3(b).

2.3.1.7 Rhythm

Regular rhythms were found to suggest happiness, dignity, majesty and peace whereas irregular rhythms may emote amusement, uneasiness and anger. A mix of rhythms could suggest joyfulness. Flowing rhythms could be perceived as happy, graceful, dreamy and serene.

2.3.1.8 Timbre

Musical tones that have a higher concentration of high harmonics (sharp) may be associated with boredom, pleasantness, happiness or sadness. Tones with a higher concentration of low harmonics (soft) may be said to be tender or sad. Experimental results with regard to orchestration have found that the sung voice may express sadness with most effect, fear by the violin and anger by the timpani. An interesting slant on the study of emotion in music is described. Western listeners were played Hindustani music and perceived that strings were associated with anger and the flute with expressions of peace.

2.3.1.9 Musical Form

Musically complex pieces with regard to harmony, melody and rhythm may suggest tension and sadness. Less tense emotions along with sadness may be associated with less musically complex pieces.

2.3.1.10 Interaction between compositional transformations

The influence of a compositional transformation on the emotion communicated to the listener is dependent upon how it is combined with other compositional cues as discussed previously in section 2.1. Opposite emotional responses perceived from the same compositional transformation could be explained by considering the effect of all the other factors contributing at the same time.

2.4 Expanded Lens Model

An expanded version of the Lens Model is presented by Juslin and Lindström (2003) and is shown in figure 4. This extension to figure 2 includes composer cues along with performer cues. The aim of this model is to investigate the interactions between the composer and the performer as shown by the addition of interaction cues to the model. Recent research (Juslin & Laukka, 2004) has shown that listeners' judgement of emotional expression can be found to be due to the individual cues identified *and* in a smaller contribution, the interactions between composed and performance cues, although not as much as has been suggested by Gabrielsson & Lindström (2001).

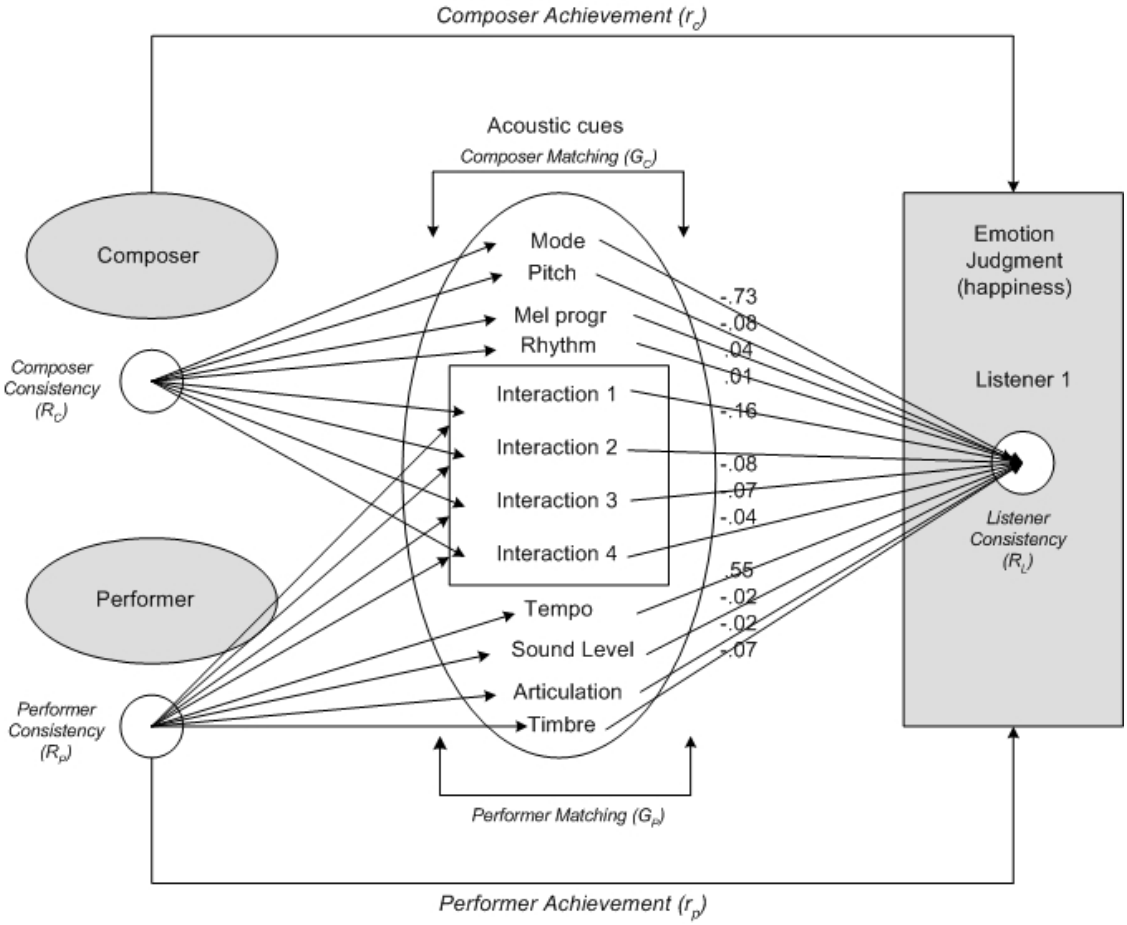


Figure 4: Extended lens model of musical communication of emotions (Adapted from Juslin & Laukka, 2004)

The values included in the figure for each individual cue refer to the beta weights of a simultaneous regression analysis (Juslin & Linstrom, 2003). For analysis in the context of this project, the values can be considered as a weighting of importance for each individual cue. The fact that a value is positive or negative is unimportant. The closer the value to 1 or -1, the higher the importance of the cue in communicating the desired emotional expression of the composer and the performer to the listener. Referring to figure 4, it can be seen that mode and tempo stand out significantly with values of -.73 and .55 respectively, as the highest weighted individual cues. This means that a major mode with a fast tempo are the most important cues that are to be used when communicating an emotional expression of happiness to a listener. Extended Lens Models for other emotional expressions are not in print currently, which is a shame, as these would be a *very* useful addition to research in this area. In the future, the article by Juslin and Linstrom (2003) may be in print.

2.5 Algorithmic Composition

Algorithmic music composition has existed for many years and has been exploited by composers such as d'Arezzo dating back to 1026. Computer based algorithmic composition techniques have been adopted by contemporary composers in a variety of ways. Stochastic algorithms, using the laws of probability, have been extensively used as decision-making tools in the compositional process. Probabilistic distributions define how likely a certain result is to come up. Common implementations of distributions include probability lookup tables of various shapes (e.g. bell curve, exponential, U-shaped). Markov chains provide another decision-making tool having the property that enables memory of past events to influence the outcome of future events. It has been shown that Markov chains can be well suited for rhythm selection (McAlpine et al, 1999) as rhythmic lines often exhibit semi-regular behaviour.

2.6 Score analysis

It has been established that there are many existing implementations of musical score analysis systems (Temperley, 2004a) that extract useful data such as beat position, beat strength, harmonic progression & key. An implementation of the score analysis system is in c code by Temperley is available from his ftp site (Temperley, 2004b).

2.7 Computer controlled music performances

Work on creating a rule set that would create a musically acceptable performance from a defined score has been carried out (Friberg & Sundberg 1986, Friberg 1991, 1995). The rules model elements of performance such as phrasing, articulation and intonation. The sound parameters that are altered by these rules include duration of tone, off-time duration, relative sound level in dB, vibrato amount and relative frequency deviation in cents. For a full description of the performance rules, refer to Friberg (1991, 1995).

2.7.1 Director Musices

The performance rule set has been implemented in the software program Director Musices (DM) (Friberg et al, 2000) that allows the rendering of different emotions from the same musical score. The rules have been designed to be independent of musical style and can generate musical performances that communicate differing emotional qualities such as happiness, sadness or anger. A musical score containing a list of track objects (melodies) that contain segments (a note or chord) form the input to DM. The rule set is applied to the score from a rule palette, and is controlled in magnitude by *k*-values for each rule that are set in DM. Note durations are specified in milliseconds and amplitudes in dB to allow for differences in synthesisers. The DM program with full manual is available for download (Friberg 2004a). Note DM is not a real-time implementation of the performance set

2.7.2 pDM

The pDM application developed by Friberg (2004b) implements real-time control of the performance rule set in the Pure Data (`pd`) environment (Puckette, 1996). Appendix K provides an introduction to the `pd` environment. The rules are pre-processed in DM resulting in an output score file in pDM format. This score contains all of the numerical rule deviations for each note, such as tempo, sound level and articulation plus the

nominal values of the original score. The pDM format is a *.txt file format, where each line comprises of a command with a delta time at the start of each command specifying the delay in execution compared to the previous line. The score is played using a player written in `pd` that allows the real-time control of the performance rule set. The score file is parsed using the *qlist* object in `pd`. Section 4.5 describes the rule application in more detail.

User control is applied either directly by manipulating *k* values to control the amount of each rule deviation or through the use of a mapper. There are two types of mapper, the emotional expression mapper and the musical expression mapper. The emotional expression mapper is a 2D Activity – Valence control space as illustrated in figure 5, where activity is related to high or low energy and valence is associated with positive or negative emotions.

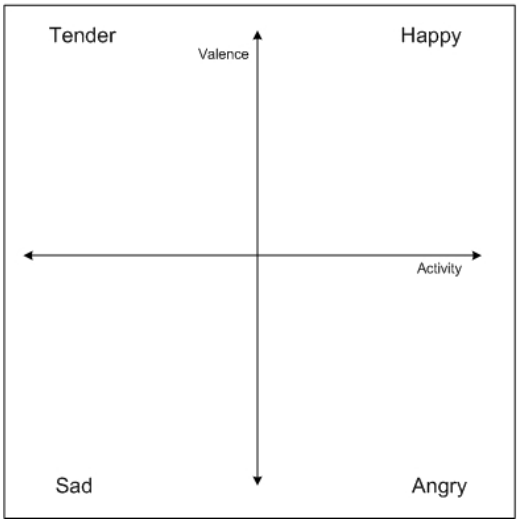


Figure 5: Activity-Valence Space (Friberg, 2004b).

Seven rules from the performance rule set that lend themselves well to real-time manipulation, are controlled in this space. Each of the four corners of the space have *k* values assigned to them and then intermediate values are interpolated according to rule selection weights to give a 3D plane of values. The musical expression mapper is a similar concept to the previous space using the same methods for generating *k* values. Gesture interfacing is also provided aimed at providing fine control over performance parameters allowing the score to be 'conducted' in real-time.

2.8 Areas still to be explored

Section 2.3 presents an extensive summary of research carried out investigating the compositional techniques that can be used to communicate different emotions. It is however not exhaustive, there are significant omissions in research into the effects of interaction between techniques and transitions between emotional states. Two extremes are considered (e.g. happy vs. sad) not taking into account any intermediate values. Does this imply that there is a straightforward interpolation between states? A non-linear 3D

plane similar to that implemented in pDM may be a solution to defining intermediate states in emotion.

The real-time control of performance parameters using pDM have been discussed in section 2.7.2. There have however been limited attempts to implement a system to enable real-time control of compositional parameters to communicate different emotional qualities.

The masters thesis "Interactive Music for Computer Games" (Hobbis, 2003), establishes a system that can process pre-composed motives and produce generative composition in the `pd` environment. The generative composition algorithm is simple and is limited to shifting the note pattern by an integer amount while maintaining the same rhythm pattern. Limitations highlighted include problems with musically unnatural results when moving from one area of interactive input space to another (e.g. 'friends' to 'danger'). The approach taken involved using two pre-composed pieces representing each state and attempting to merge them together.

Microsoft DirectMusic allows music and sound effects to be played in applications. Microsoft DirectMusic Producer (Microsoft, 2003) is a tool that enables composers and sound designers to create interactive audio in the DirectX (Microsoft, 2005) format using DirectMusic. Sections of MIDI and audio can be assigned to states in applications to allow particular themes to be played at key moments. Multiple layers of MIDI and audio are allowed. Certain levels of variability are applied to real-time playback such as note timing and tempo changes along with the ability to manipulate a pre-composed melody around a pre-defined chordmap and style.

2.9 Discussion of Survey

Table 1 has presented Hevner's results from studies carried out. There are some points that are worthy of comment. The study into manipulation of melodic direction produces low relative weights (if any) for ascending or descending melodies. This suggests that there may be certain conflicts in the communication of specific emotions through melodic direction and confirms later research carried out that it has little (if any) effect on perceived emotional expression. Melodic direction has therefore been discarded as a tool that could be used as an effective way of communicating expression. Mode, tempo and pitch all have high relative weights suggesting that these musical factors play a large part in determining the emotional expression of a score. Rhythm and harmony also have significant relative weightings. Happy and sad emotions both have similar relative weightings for important structural factors to influence the perceived musical expression, but at opposite ends of the valence scale (see figure 6). For example, a major mode is identified as very important for happy emotions whereas a minor mode is considered important for sad emotions.

There is a lack of two of the so-called basic emotions, tender and angry. Using data collected from later studies (Gabrielsson & Lindström, 2001), a similar observation can be made as with the happy/sad emotions. Tender and angry emotions lie at opposite ends

of the activity scale. For example, rhythms that are irregular/rough have been found to express anger whereas flowing rhythms have been found to express tenderness. Structural factors that have been found to be important to communicate the perceived expression have been mapped to the Activity-Valence space in figure 7.

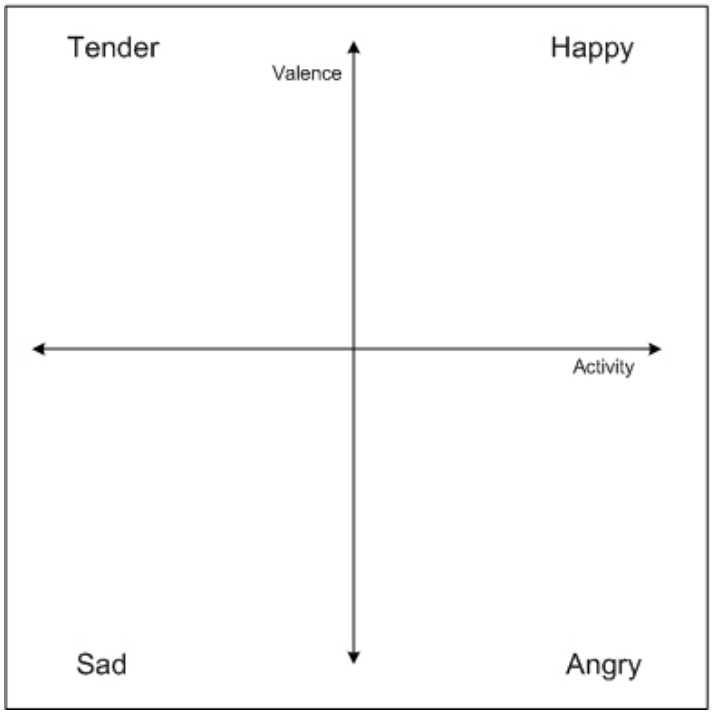


Figure 6: Activity-Valence control space with superimposed Activity and Valence axis

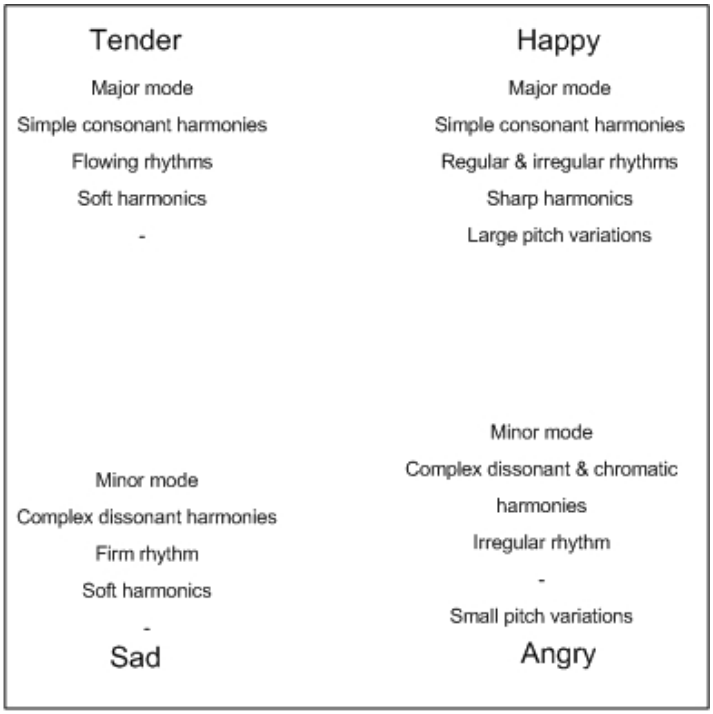


Figure 7: Activity-Valence control space with key structural factors mapped to each corner

The above figures show a good correlation between positive and negative values on the Activity-Valence axes and the key structural factors mapped to each corner. A concise model showing the nature of the key structural factors found to be significant to communicate an emotional expression has therefore been created.

2.10 Summary of Survey

Based on the research carried out, scope exists to develop a system that gives real-time emotional feedback to a user using compositional tools such as those summarised in section 2.3 and figure 7 above, using rhythm, timbre and harmony rule sets. These structural factors can be manipulated in real-time in a musical context if they were interpolated within the Activity-Valence control space. This system could then be integrated with the current implementation of pDM to allow the real-time control of performance and compositional cues together using any mapping system required. There would then be an interaction between the composed structural cues of a piece of music and the performance cues of a performance. This aim, along with investigation into how changes in key structural factors identified can be implemented, forms the focus of the project. Section 3 now presents the main project aim and objectives.

3 Aims and Objectives

3.1 Aim

To design and build a software application that allows real-time control over the structural factors of a composition, identified in section 2, in a musical context to communicate different emotional states.

3.2 Objectives

1) Structural changes shall be applied to a pre-composed score in real-time, and be controlled by the user input.

2) User input will take the form of the Activity Valence control space.

3) Musical output shall demonstrate the structural qualities defined in each corner of figure 7 when the user input position is located at co-ordinates (-1,1), (1,1), (-1,-1) and (1,-1).

4) The input score shall contain a melody line with a specified harmonic progression, and a percussion part.

5) An accompaniment and bass part shall be generated using the harmonic progression specified in the score as a guide. These parts should reflect the structural qualities defined in figure 7 with regard to the mode, harmony and pitch variations.

6) Structural changes to the musical output shall be applied in a gradual manner between the basic emotions when user input position is changing.

7) The system shall be designed to allow for integration with pDM to create a real-time musically expressive sequencer that models the performer along with the composer.

3.3 Block Diagram derived from Objectives

Figure 8 illustrates a block diagram derived from the objectives stated in section 3.2. This diagram shows the main functionality block that the system is required to perform.

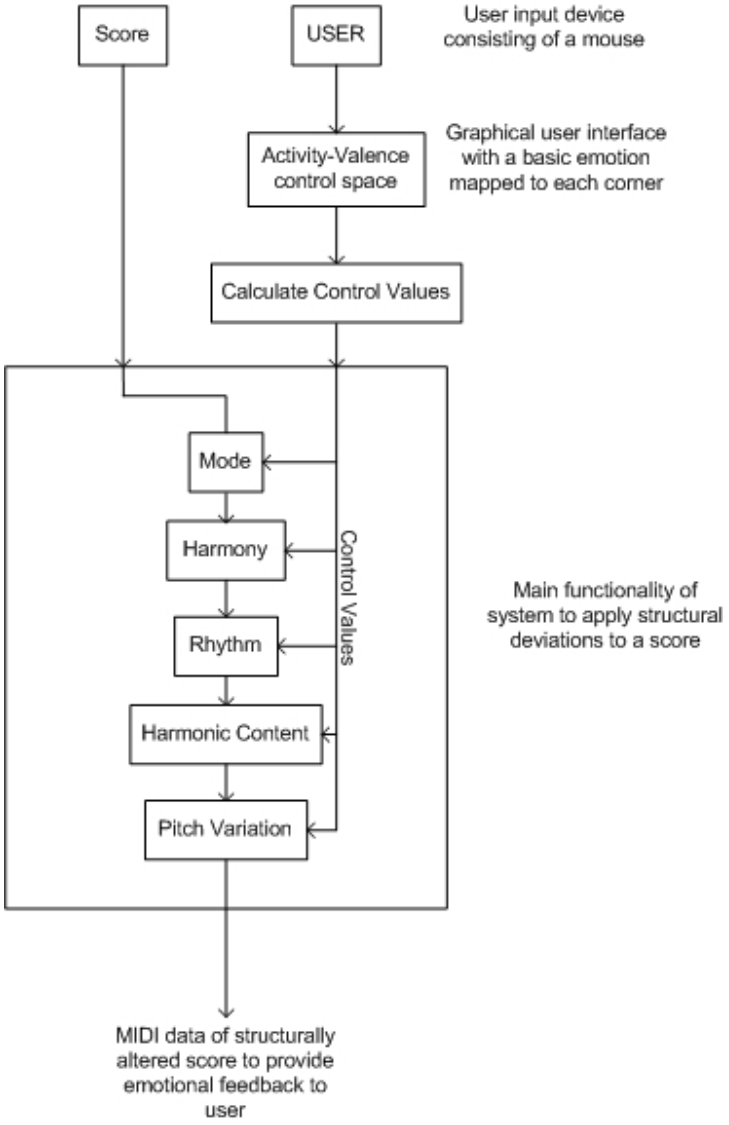


Figure 8: Block Diagram of system derived from project objectives

4 Theoretical Studies

This section presents research carried out during the project to find an effective way of achieving the type of structural changes mapped to the Activity-Valence control space illustrated in figure 7. Knowledge essential to the project is also presented such as the internal structure of pDM

4.1 Harmony

Figure 7 shows that simple consonant harmonies express tenderness and happiness and complex dissonant harmonies express sadness and anger. The following section explains these terms and describes how the concept can be used to express different emotions.

4.1.1 Consonance and Dissonance

The psychoacoustic concept of consonance and dissonance refers to the pleasantness of a sound (Howard & Angus, 1998). A concept using critical bands relating to the basilar membrane is commonly used to define whether a sound is consonant or dissonant in nature. The basilar membrane is part of the cochlea found in the inner ear. The function of the membrane is to carry out frequency analysis of sounds received by the human ear. If a pure tone is heard, a small area of the basilar membrane will be stimulated relating to the frequency of the input tone. This displacement bends hair cells distributed along the membrane that form the auditory nerve. The properties of the basilar membrane define the accuracy in which humans can discriminate between two separate tones. If two are sounded simultaneously, there will be two displacements resulting from each tone. If the displacements are located close together on the membrane so that they interact to produce a single peak displacement, a rough sound will be heard. This is because the basilar membrane cannot distinguish between the two separate tones. If the tones are almost equal, a beating effect will be heard. The point at which two separate smooth tones become clearly heard is used to calculate the critical bandwidth (CB). The CB is the frequency difference between these two tones. Scharf (1970) gives a formal definition: "The critical bandwidth is that bandwidth at which subjective responses rather abruptly change"

If the frequency difference between two tones is calculated to be between 5% and 50% of the CB, the interval is said to be dissonant (Howard & Angus, 1998). The maximum dissonance occurs when the frequency difference is ¼ of the CB. To calculate the value of the CB, an equation proposed by Moore and Glasberg (1983) can be used:

$$\mathrm{ERB} = \{[6.23 \times 10^{-6} \times f_c^2] + [93.39 \times 10^{-3} \times f_c] + 28.52\}\ \mathrm{Hz} \quad (1)$$

Referring to (1), f_c represents the filter centre frequency (or fundamental frequency) in Hz and ERB represents the equivalent rectangular bandwidth in Hz. This equation is valid for ($100\mathrm{Hz} < f_c < 10\,000\mathrm{Hz}$).

The concept of critical bandwidth can be used to create dissonant harmonic intervals and harsh timbres to convey expressions of anger and sadness.

4.2 Rhythm

Rhythmic properties for each basic emotion have been proposed in figure 7. A method of manipulating a rhythm to posses these properties is needed. An established method of classifying beats is proposed by Temperley (1999) in section 2.6 of the literature review. This method is used in score analysis to extract information such as beat position and beat strength. A model called the Theory of Meter is presented as illustrated in figure 9. This consists of a metrical structure that consists of several levels of beats. Each beat is a point in time and does not therefore always represent an event. Every second beat at one level represents a beat at the next level up. The level with the fastest beats is called the lowest level.

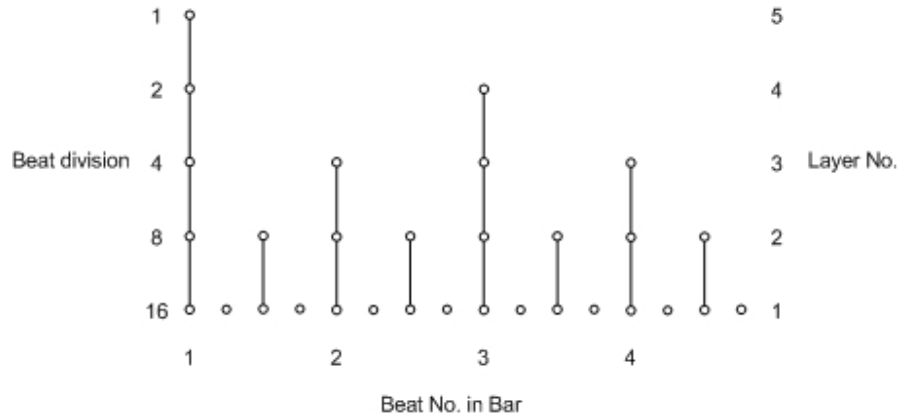


Figure 9: Theory of meter for 4/4 measure (Adapted from Temperley, 1999)

This model can be used to estimate the perceptual prominence of each position in the bar. The first beat in the bar belongs to layer 5 and is thus the most prominent beat. Layer 4 represents the third beat in the bar and therefore the next most prominent beat of the bar. This order of importance continues down to Layer 1, the bottom layer.

If there is a percussion part defined in a score, this model can be used to change its rhythmic properties by removing of layers to create less dense patterns. If the current 16^{th} note (position along the layer 1 line) in a bar is tracked in an arppegiator style arrangement, each 16^{th} note segment will have a relative layer number assigned to it. These layers can represent a weighting of importance for each 16^{th} note segment (i.e. layer 5 represents the first beat and thus the most important beat of the bar). By allowing only certain configurations of layers to be sounded, the pattern of the percussion part can be changed.

To allow for more flexibility than just effectively muting the percussion part at certain positions in a bar, the ability to choose specific instruments to sound from the part would be a useful feature.

4.3 Harmonic and Rhythmic Tension

Harmonic and rhythmic tension refers to specific ways of altering the structural content of a score to produce unexpected outcomes. Harmonic tension may be introduced to a chord progression by violating our expectations with respect to the tonic (the first note of a musical scale). Cadences provide closure to a progression through cadential formulas. A full explanation of the different types of cadences can be found in Appendix J. Different degrees of closure/resolution are provided by different cadences. Effective cadences using just major/minor triads must contain three chords (Aikin, 2004).

Major	IV – V – I	ii – V – I	V – ii - I
Aeolian	v –vi – i	^{b}VII – iv – i	^{b}VI – ^{b}VII – i
Harmonic Minor	iv – V – i	V – iv – i	^{b}VI – V - i

Table 2: Three triad cadences

Table 2 shows examples of three triad cadences in different modes, Major, Aeolian and Harmonic Minor. Each progression reads from left to right, starts with an *antepenult*, then a *penult*, and finishes with a *final* triad. The numerals represent the root of the chord related to the tonic of the current key. The classical *IV – V – I* and *ii – V – I* cadential chord progressions firmly establishes the I chord as a home position of rest and stability. This effect is most prominent when the chords used are played in root position. If inverted triads are used, the effect is not as potent and the result is more neutral.

Two-chord cadential sequences are shown in table 3. These sequences, consisting of a *penult* followed by a *final* chord are an extension beyond simple triads. The penult is usually a dominant 7^{th} or an augmented 6^{th}.

Major	V^7 - I
Aeolian	$^{b}VII^7$ - i
Harmonic Minor	V^7 - i

Table 3: Two chord cadences

Harmonic tension can also be provided by the juxtaposition of consonant and dissonant sounds. It is usual that all dissonances are resolved to a consonant conclusion.

Rhythmic tension is governed by the layering of rhythms and polyrhythms and can be relaxed by a return to a regular pulse. The effect of rhythmic dissonance can be created by having two conflicting rhythms. Tense situations can be created when the rhythmic component of a score is in contradiction to the harmonic component. Changing of cadences are not implemented in the project, but the details are included for the reader as information regarding voicing of chords, described above, is used in the accompaniment part.

4.4 Mode

Changing mode from major to minor is identified as being an important structural change necessary to effectively change the perceived expression towards the lower half of the

Activity-Valence control space. The harmonic minor mode contains an augmented second found between the sixth and seventh degrees (see table 4).

1	2	$^{b}3$	4	5	$^{b}6$	7	Notes
i	ii^{o}	$^{b}III^{+}$	iv	V	^{b}VI	vii^{o}	Chords

Table 4: Harmonic minor mode scale

This scale mode offers powerful tonality, possibly more "negative" sounding, not offered by the Aeolian mode below. The Aeolian minor scale, shown in table 5, offers a relaxed and softer sound than the harmonic minor.

1	2	$^{b}3$	4	5	$^{b}6$	$^{b}7$	Notes
i	ii	^{b}III	iv	v	^{b}VI	^{b}VII	Chords

Table 5: Aeolian minor mode scale

4.5 pDM

As integration with pDM (Friberg, 2004) is stated as objective number 7 in section 3.2, knowledge of the internal operation of pDM is required. As described in section 2.7.2, pDM is a MIDI score player with real-time control of the KTH performance rules.

The rules are applied in DM resulting in an output score file in pDM format. This score contains all of the rule deviations, such as tempo, sound level and articulation plus the nominal values of the original score. The pDM format is a text file in a *.pdm format, where each line comprises a command with a delta time at the start of each command specifying the delay in execution compared to the previous line. The score file is parsed using the `qlist` object in `pd`. Performance rule deviations are present in the score in ΔT (DT), ΔSL (DSL), ΔART (DART) where each means relative tempo deviation, sound level deviation, and articulation in terms of pause duration respectively.

The score files begin with a header section containing information such as score name and version. Each command line begins with a delta time indicating the delay in milliseconds relative to the previous command, followed by an object name and data:

```
<delta time (ms)> <object name> <data>;
```

The object can be either a new note (NOTE), ΔT (DT), ΔSL (DSL), ΔART (DART) or a header command.

| means logical *or*.

```
<object name> = NOTE | DT | DSL | DART | <header command>
```

The NOTE format gives values for each note:

```
<delta time> NOTE <MIDI note number> <MIDI channel (1-16)> <velocity>
<note duration (ms)>
```

The design of pDM assumes that rule deviations apply to the subsequent note. The DT format gives the values generated by the tempo deviation rule for each note:

<delta time> DT <ΔT_1> <ΔT_2> <ΔT_3> <ΔT_4>…

In a similar fashion, the rule deviations for ΔSL and ΔART are in the format:

<delta time> DSL <ΔSL_1> <ΔSL_2> <ΔSL_3> <ΔSL_4>…
<delta time> DART <ΔART_1> <ΔART_2> <ΔART_3> <ΔART_4>…

A list of performance rules that are applied to the input score in DM is available (Friberg, 2004b).

5 Development and Design

Development environment options that are suitable to implement the stated objectives given in section 3.2 are limited. Microsoft Direct Music could be a possible development environment used. However, there is little ability to control the structural changes to a composition that have been identified. This choice of environment would also make integration with pDM difficult as it differs to the Pure Data graphical programming environment that pDM is developed in. Analysis of the block diagram of the proposed system shows that it is based around modular structure of key functions. `pd` lends itself well to such an approach as it is a real-time graphical programming environment for processing MIDI and audio, where a modular structure can easily be adopted. Fast implementations of ideas are possible leading to a quick software development cycle. `pd` is also platform independent. jMAX (Déchelle, 1998) and MAX/MSP (Puckette, 1989) are implementations of the `pd` concept in Java and originally for Macintosh respectively. These distributions could be used for the development of many of the project objectives except number 7 given in section 3.2. The objective to integrate the functionality of this project with pDM will be achievable by the use of `pd` and hence the `pd` environment stands out as the choice for development.

The direct use of certain building blocks implemented in pDM is possible for use in the system being developed for this project; therefore, the patch is developed in the `pd`-extended distribution, 0.37. This distribution includes all standard libraries and can be found on the accompanying CD.

A brief introduction to `pd` and some of the common concepts used can be found in Appendix K.

5.1 Director Musices

Director Musices (DM) is used to create a .txt (with custom extension *.pdm) score file from a General MIDI file consisting of a melody part and a percussion part. In DM, a score file in General MIDI format is loaded, a set of performance rules chosen and then applied to create a file that can be listened to. This program is used to create the test scores using the following procedure: (1) The performance parameters are reset to their nominal values. (2) The first performance rule is applied. (3) The deviations that result from the rule application are normalised and collected in a result vector. These steps are repeated for each performance rule. The score format for the application developed in this project is an extension to the format defined for pDM, and thus DM can create test scores for this application with only minor additions to the processing stage. Test scores were created using this method in *.pdm format.

The presence of the performance rule deviations in the score is for two reasons. Firstly, the current implementation of DM includes these deviations in the process of creating the test score, and secondly the integration of compositional rules and performance cue rule set will be aided by their inclusion.

5.2 Description of Score Format

A score format had to be defined so that specific data could be parsed to `pd` to provide the information necessary for the application to function as required. This format is given in Appendix G. This format is compatible with pDM as it contains the full header information required by the application plus extra object data described below.

The test score files begin with a header section containing information such as score name, version, key and meter. Each command line begins with a delta time indicating the delay in milliseconds relative to the previous command, followed by an object name and data:

```
<delta time (ms)> <object name> <data>;
```

The object can either be a new note (NOTE), a new chord (CHORD), a new key (KEY), or a header command.
| means logical *or*.

```
<object name> = NOTE |  CHORD | KEY | <header command>
```

The NOTE format gives values for each note:

```
<delta time> NOTE <MIDI note number> <MIDI channel (1-16)> <velocity>
<note duration (ms)>
```

The CHORD format gives data for chord root and tension:

```
<delta time> CHORD <chord root (Bb)> <chord tension (Maj7)>
```

The KEY format gives data for the current key and mode:

```
<delta time> KEY <key tonic (Bb)> <mode (Maj/Min)>
```

The current note names and chord tensions supported are given below in tables 6 and 7 respectively.

C	C#	Db	D	D#	Eb	E	E#	F	F#	Gb	G	G#	Ab	A	A#	Bb
B	Cb															

Table 6: A list of note names supported

Chord Tension	Full Chord Name
maj	Major
min	Minor
maj6	Major 6th
min6	Minor 6th
maj7	Major 7th
min7	Minor 7th
dom7	Dominant 7th
dim7	Diminished 7th

Table 7: A list of Chord Tensions related to their full name

Performance rule deviations are also present in the score as described in section 4.5 to allow compatibility with pDM.

5.3 System Design Block Diagram

Figure 10 presents a full system design diagram illustrating the main modules and data paths that are to be developed in the application. All information that is parsed to `pd` from the input score is shown in relation to the sub-patch that requires the information.

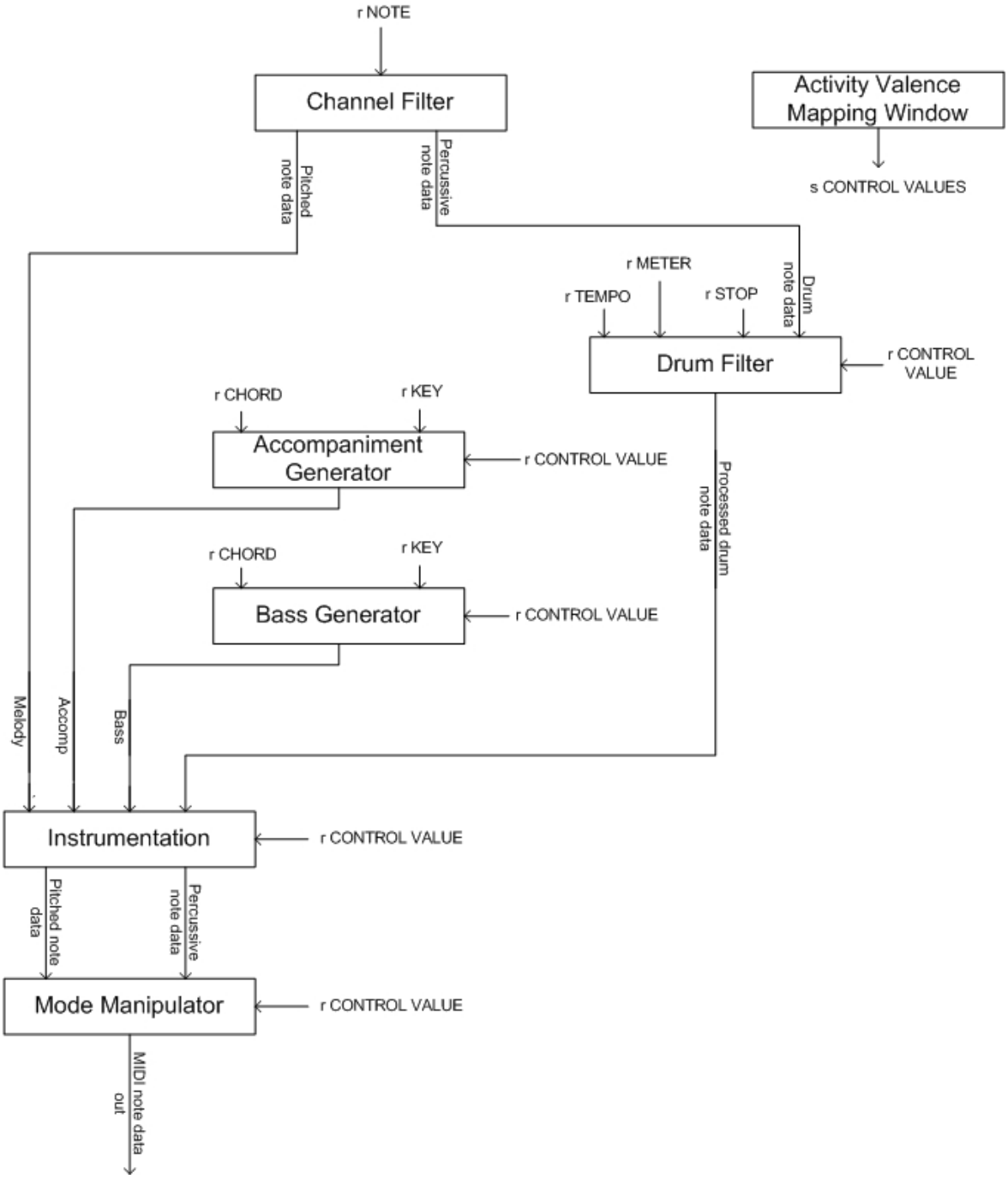


Figure 10: Full system design block diagram

A description of the functionality of each of the blocks shown in the figure is given in the following sections. The small *s* and *r* characters in figure 10 refer to send and receive actions for the data that follows.

6 Extracting Useful Data to PD

The following sections describe the implementation of the system in the Pure Data development environment. Full graphical representation of every sub-patch can be found in Appendix C. Control values are used throughout to control the amount and type of structural changes applied and are described in section 11.

6.1 sequencer Sub-patch

The sequencer sub-patch contains all the modular functionality shown in figure 10 except the `activity_valence_mapping_window`. The *qlist* object is used to parse the `.pdm` text file into `pd` to allow the score information to be used. The implementation of this object is shown in figure 11. *qlist* generates messages at the time they are meant to be played and can therefore be replaced with real-time input data. There is no look ahead or look back facility.

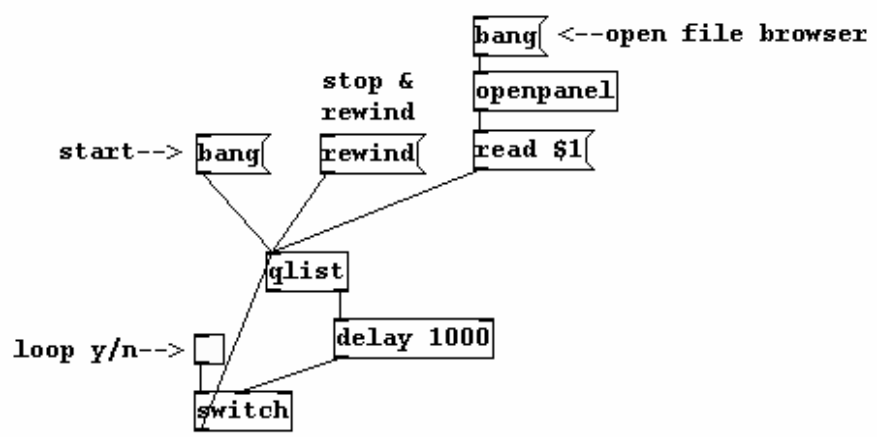


Figure 11: *qlist* object and methods accepted.

The object accepts several methods that provide simple sequencing functionality. The *openpanel* object displays a file browser on screen and sends the file selected to its outlet. The read method receives the file path that *qlist* should open and sends it to the leftmost inlet. A bang message sent to the same inlet runs the sequencing function of *qlist*. The rewind message stops the *qlist* object and rewinds to the start of the current file. An option to loop the *qlist* back to the beginning once the end of the file is reached can be implemented using a *switch* object. A bang message is sent to the right outlet of *qlist* when the end of the file is reached. This bang message is delayed in the pipe object by an amount specified in its creation argument and appears at the outlet of the *switch* object if the toggle box is checked to loop the sequence. This bang restarts the *qlist* object.

Each line of the text score has a delta time that represents the delay in milliseconds relative to the last line. Each delta time is followed by a symbol stating the destination object in `pd`. Any numbers or symbols that follow the destination object are outputted as

a list. For example `1000 NOTE 60 1 100 2` would send 60, 1, 100 & 2 to the object 'NOTE' 1000 milliseconds after the last command line was read.

The *sequencer* sub-patch contains the main functionality for the application, and is structured in a similar manner to the block diagram of the system. All note data is processed from this window.

6.2 channel_filter sub-patch

The melody and percussion parts are separated in the `channel_filter` sub-patch as they are processed independently. This sub-patch uses MIDI channel number to test the incoming note data and decide where it should be sent. MIDI channel 10 contains percussion instruments and therefore any note data addressing channel 10 will always be a percussion part. All data from MIDI channel 10 is sent to the right outlet of the sub-patch. All other note data is sent to the left outlet. The implementation of the `channel_filter` is not ideal in terms of maximum efficiency as there are many duplicates of the *match* object. This is due to the lack of an outlet that contains all input data that does not meet the matching criteria specified in the objects' argument.

Another solution to the problem the *match* object created has been considered. Custom objects can be designed for `pd` using the C++ coding language called externals. These are compiled to produce a `.dll` library file that is placed in the `pd` bin directory ready to be called from the graphical environment. A simple design for an object that provided an outlet that contained all input data not meeting the input matching criteria could have been developed. Development of externals in this project has been limited due to the issues with compatibility with other systems. Patches designed using the standard release libraries are compatible with any system and platform with `pd` installed. If a custom library were used with a `pd` patch, it would have to be compiled differently for each of the platforms supported by `pd` and then supplied with the patch. The decision was therefore made to use only the standard libraries for the duration of the project. It has been noticed that documentation providing information regarding the production of externals for `pd` is very limited, and for this reason, there is a brief guide given in Appendix E on the production of externals.

6.3 Note Output

The output of processed note data is sent to the *noteout* object.

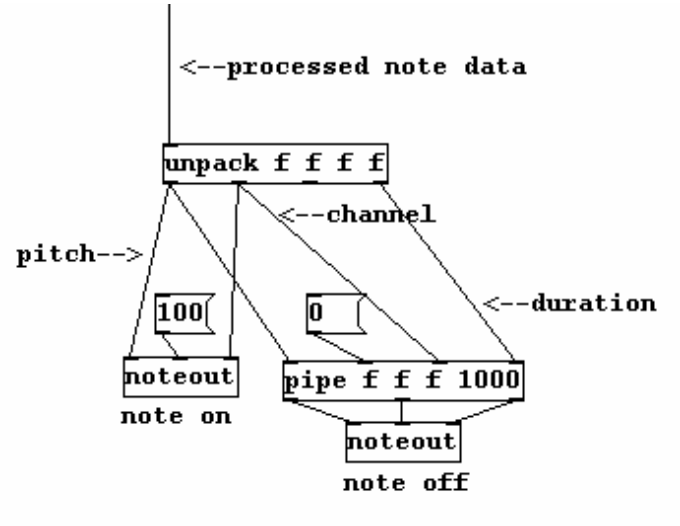


Figure 12: Method of outputting note data

Two *noteout* objects are shown in figure 12. The first produces note on messages and receives MIDI pitch, a fixed velocity of 100, and the MIDI channel number. Note off messages are created by using the duration value for each note to delay a copy of the original note on message with the velocity set to zero. The delay is achieved by using a *pipe* object. The far right inlet to the object is used to set the delay time. A creation argument may be specified in the *pipe* object as shown in figure 12, specifying the initial delay of the pipe. This value is overwritten and replaced when a value is present at the far right inlet. An essential feature is the facility to reschedule the delay time. This means that *pipe* schedules its output as soon as an input message is received, allowing delay time to be changed at runtime without affecting the data that has already been scheduled to appear at the outlet. This architecture is an efficient way of creating note on and note off messages, allowing a note to be specified in the score in terms of length and requiring no note off messages. MIDI notes should not be left stuck 'on' with this design as all notes have a note off message scheduled.

7 Generation of Accompaniment & Bass Parts

7.1 Accompaniment Part

The chord and key objects read from the score file provide the accompaniment generator module with the necessary information to produce a chord backing. This information is also used to generate the bass part in the Bass Generator (`bass`) module.

7.2 acco sub-patch

7.2.1 Functionality

The `acco` sub-patch assigns a MIDI pitch to the tonic of the current key. This pitch is set as a centre pitch that all incoming CHORD root notes are pitched around. In effect, the tonic of the key is middle weighted as shown in figure 13.

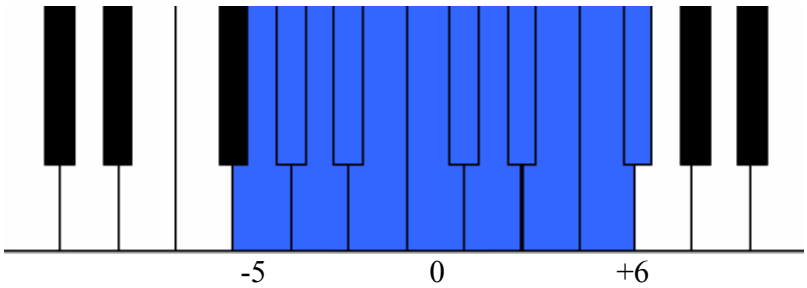


Figure 13: Tonic middle weighted at position zero.

7.2.2 Implementation

The format of CHORD has been described in section 5.2. A full description of the score format can be found in Appendix G. Data in this format is received by the CHORD object and unpacked to extract two symbols and a number. The first symbol represents the root of the chord. Musical convention allows for more than one name for each pitched note, e.g., Gb refers to the same note as F#, assuming that the two tones are tuned to the exact same pitch. Flexibility in note naming is allowed for by routing the chord root symbol through a *select* object, shown in figure 14, that is designed to combine common note names used to describe the same musical pitch so that the correct note is selected.

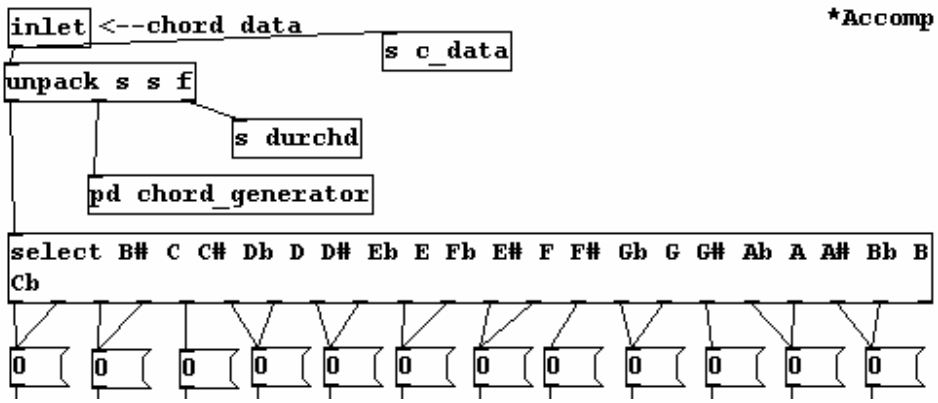


Figure14 : Select object to combine common note names

The KEY object receives two symbols when the score is read, the tonic and the mode. The tonic is routed through an identical copy of the *select* object shown in figure 14, used to accommodate the fact that musical notes can be described by more than one name. If the tonic is C (or indeed B#), a set of numbers stored in a list is sent to initialise C as position zero for the accompaniment generator. Intervals up-to 6 semitones above the tonic are defined and down to 5 semitones below. This means that chords with root notes up to F# will be pitched above C, and chords with roots from G to B will be played below C.

Each time the CHORD object receives new data from the score, the appropriate root note name is selected as described, then combined with the scale position to give the deviation in semitones the chord root is from the key tonic. The key tonic has been represented by zero up-to this stage. The key tonic is converted to MIDI pitches, where middle C = 60. The tonic MIDI pitch is added to the number representing the semitone deviation from the tonic to give the actual pitch of the root of the current chord.

Notes are added around the root note to build up a chord to be played. The number of notes and their pitches are determined by the `chord_generator` sub-patch.

7.3 chord_generator sub-patch

7.3.1 Functionality

The chord generator is designed to select the most suitable voicing of a chord according to control values derived from the user input in the Activity-Valence control space. There are four harmony types mapped to each corner of the Activity-Valence control space. For happy, all chords are triads (major or minor) and played in root position as this provides the feeling of stability as described in section 4.3. Tensions such as dom7 are dropped. The pitch of the root of the chord is determined as described above with the key tonic being middle weighted. For tender, all chords are played as specified in the score, for example C Maj7. Inversions of the chord are selected so that the current chord is voiced as close to the tonic root as possible resulting in a narrow pitch range for the accompaniment as specified in figure 7. The register that the chords are sounded in is the 4^{th} and 5^{th} keyboard octave. For sad, the chords are voiced in a lower register than they

are sounded for the tender emotion, still with small pitch variations between chords. For angry, an even lower register voicings are selected with the addition of a Perfect Fourth intended to sound dissonant due to tones being sounded that are less than the critical bandwidth (CB).

The CB can be calculated to ensure that dissonant and harsh timbre tones are produced as specified in the mapping of structural factors to emotions in the Activity-Valence control space. Using the equal tempered chromatic scale taking A4 = 440Hz (American Standard Pitch), each successive pitch is related to the previous by a factor of the twelfth root of two. An octave corresponds exactly to a doubling of pitch. Intermediate pitches are found by multiplying (dividing) a given starting pitch by as many factors of the twelfth root of two as there are steps up (down) to the desired pitch. For example the G above A4 = $\sqrt[12]{2}\text{^}10 * 440 = 783.99Hz$. For a G minor triad in the fourth octave the fundamental frequencies correct to 4 d.p are as follows:

G4 = 391.9954 Hz
Bb4 = 466.1637 Hz
D5 = 587.3295 Hz

Adding a Perfect Fourth of C5 of fundamental frequency = 523.251 Hz. The CB at the fundamental frequency of C5 = 79.09 Hz (3 d.p). Subtracting the fundamental frequency of D5 from C5 = 65.079 Hz. This value is within the CB and produces a harsh tone. Whether the interval can be judged dissonant or not as the frequency difference between the notes is greater than 50% of CB is a question that remains to be answered. Experiments have resulted in a Perfect fourth being personally the preferred interval to add to a minor triad as opposed to a major second. This is due to the chord sounding too complex and ‘muddy’ with an added major second.

For example, figure 15 shows that a score in the key of G Major with the following chord progression would be voiced in the following way for each of the basic emotions:

Chord Progression: *G Major -> A min7 -> B Minor -> C Maj7*

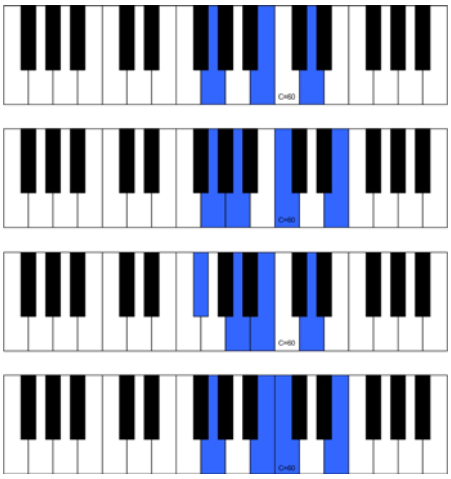


Figure 15(a):Tender Voicing

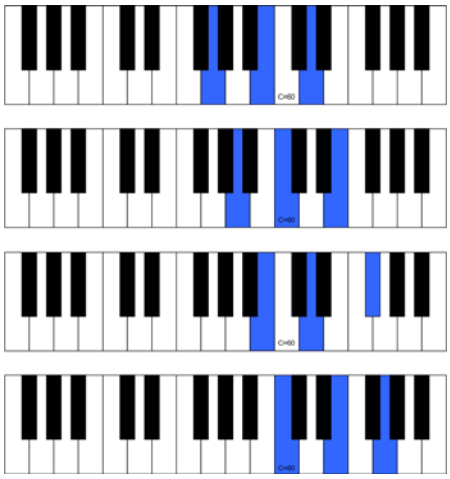


Figure 15(b):Happy Voicing

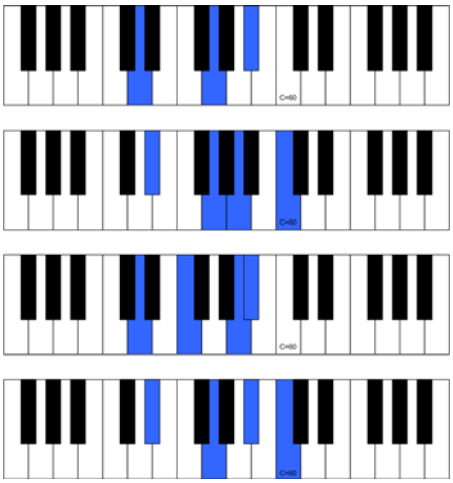


Figure 15(c):Sad Voicing

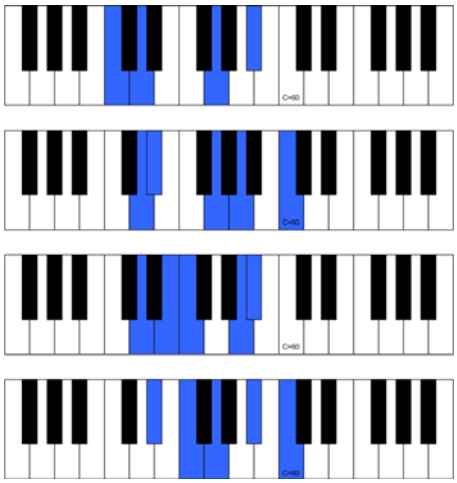


Figure 15(d):Angry Voicing

7.3.2 Implementation

The type of the current CHORD (e.g. Major 7th) is given in the second symbol of the CHORD command line as described in section 5.2. The *select* objects illustrated in figure 16 select either major or minor triads to be played, dropping all other tensions, or play the tensions as scored. The user position in the Activity-Valence control space determines which *select* object is used. Section 11.1.3 provides information regarding mapping of harmony control values.

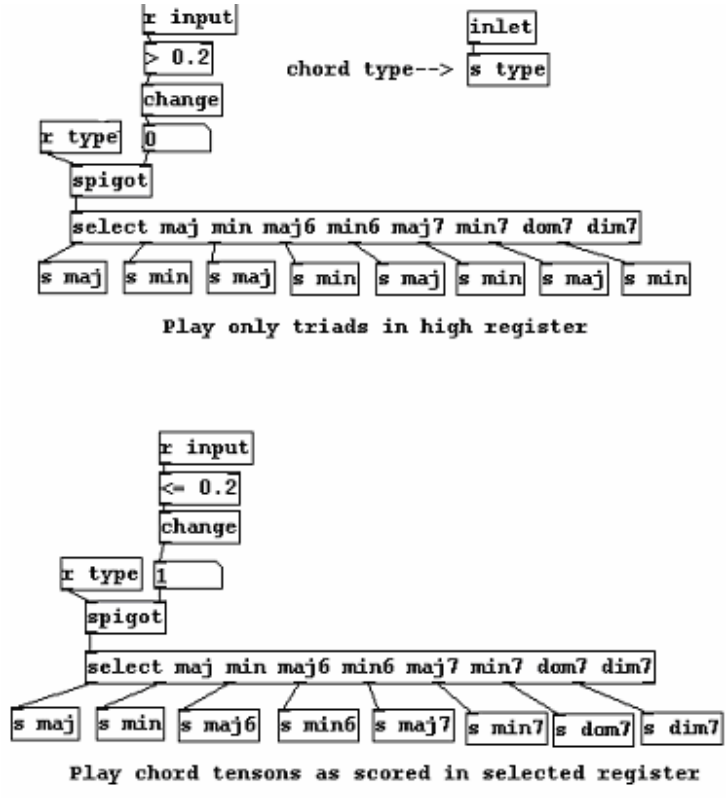


Figure 16: Select objects used to select type of chord tension to play

The *select* object sends a bang message to the appropriate chord tension list as shown in figure 17 using send and receive objects to keep the visual appearance of the sub-patch clear.

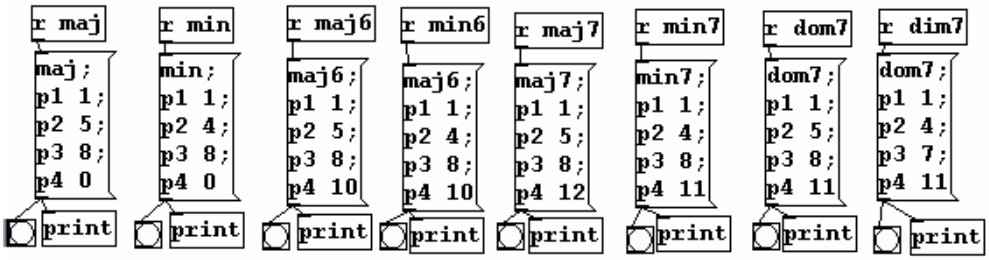


Figure 17: Lists containing elements to represent the semi-tone intervals to make up a chord

There are five levels of data stored in each list shown. The first is the abbreviated name of the chord tension. The following four values represent pitches to be played to make up the specified chord. List element 'p1' represents the root note of the chord as calculated in section 7.2.2. List elements 'p2' and below represent the further pitches above the root note to be played. If only three notes are needed to make up a chord such as a major triad, the element 'p4' is set to zero. All of the list element values are received by the `acco` sub-patch to be transformed into MIDI note on messages.

Further sets of lists contained in the `chord_generator` sub-patch apply inversions to the current chord, dependant upon the user position in the Activity-Valence control space. The four lists shown in figure 18 are structured in a similar way to the lists described above.

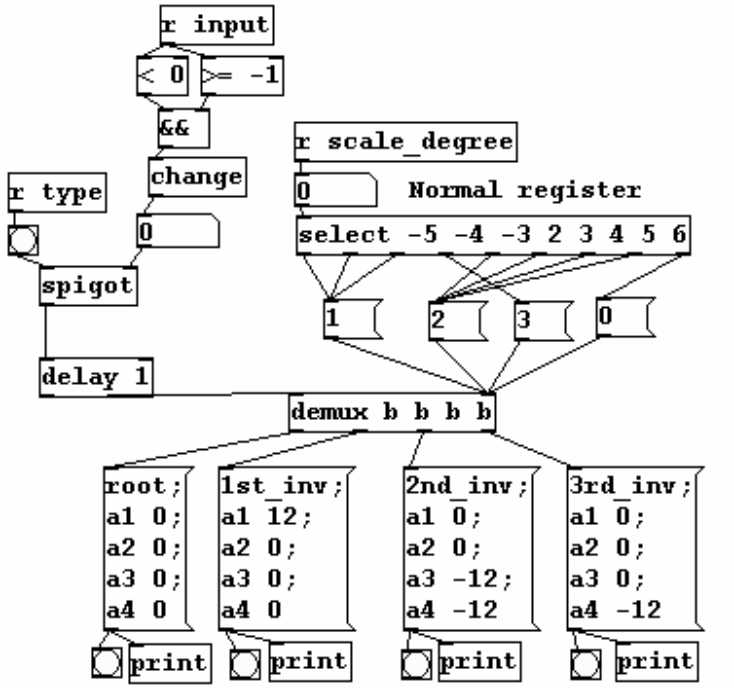


Figure 18: Lists containing octave transpositions to play chords in different inversions

The list elements 'a1' to 'a4' relate to list elements 'p1' to 'p4' from the previous lists shown in figure 17. They contain either a zero, that has no effect on the note pitch, a 12, that transposes the note up one octave, or a -12, that transposes the note down an octave. The effect is to change the inversion of the chord played. The type of list selected and therefore the type of inversion played is dependant upon the position of the current root note related to the key root pitch. The *select* object provides this functionality. For example, chords with root notes lower than three semitones from the key root note are played in the first inversion, where the root note is transposed up an octave. All of the list element values are received by the `acco` sub-patch, that adds the values to the values of the chord pitches calculated. The *delay* object shown in figure 18 is set to a delay of 1ms to ensure that the inversion is applied to the correct chord, as problems with the previous chord being affected were encountered.

Originally, the score format contained no value for the duration in ms of each chord. This decision was made as the *pipe* object has a flush method to empty any data stored in the line immediately. This design was intended to be used to trigger note off messages when the next chord arrived from the score. However, this method, although listed in the `pd` documentation, fails to work in the context that it is described. This meant that the decision had to be made to add duration values to each chord in the score format. These durations are used to trigger note off messages in the same way as all other note data.

7.4 Bass Part

7.4.1 bass Sub-patch

The root of the chord is used to generate a bass part. The pitch of the bass voice is an octave lower than the pitch that is set as the current chord root. A note is played when a new CHORD object is read in the score. The duration of the note is as specified in the CHORD object in the score. A picked bass voice is used corresponding to MIDI program number 35 to produce a generic bass sound.

8 Manipulation of Rhythm and Percussion

8.1 drum_filter Sub-patch

The `drum_filter` sub-patch selects which layers of the beat to play and which instrument groups to play. It contains three sub-patches, `metronome`, `layer_indicator` and `bypass_rhythmic_variations` respectively. The first two sub-patches provide the core functionality of the `drum_filter`.

8.2 `metronome` Sub-patch

The `metronome` sub-patch calculates how many 16th beat notes are in each bar and receives the length of each 16th beat in milliseconds. This information is then used to increment a counter representing the current 16th note position in the bar.

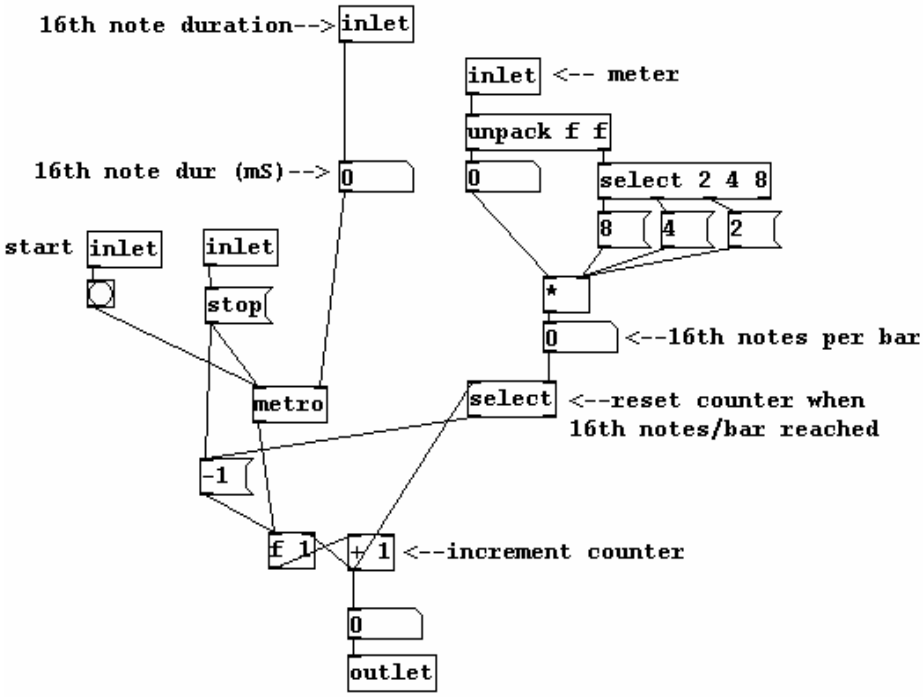


Figure 19: Metronome sub-patch

The format of METER is as follows:

```
delta time METER <mult> <div>
```

where `mult` represents the numerator, and `div` represents the denominator of a time signature. If the METER returned a value of `<4> <8>` for example, in musical terms this represents four quaver beats per bar. This therefore results in eight semiquaver (16th note) beats per bar. The *select* object in figure 19 calculates the number of 16th notes in each bar of the input score. An `<8>` for the divisor triggers the message box with the

value two stored inside. The value two represents the number of semiquavers per quaver. The triggered value is then multiplied by the number of beats in a bar (`<4>` in this case) to produce the correct result.

8.3 layer Indicator Sub-patch

`layer_indicator` receives the current 16th note position from metronome and selects which rhythmic layer, 1 though 5, the current position belongs to using an instance of the *select* object. This *select* object implements the Temperley Theory of Meter shown in figure 9. Once the process of allocating a rhythmic layer to the current 16th note has been completed, the corresponding rhythmic layer sends a bang message to its' assigned outlet.

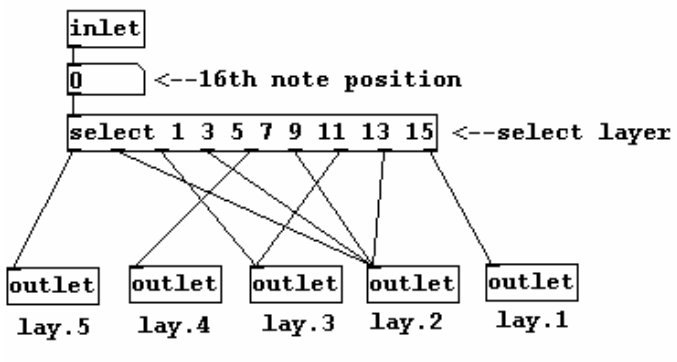


Figure 20: Implementation of Theory of Meter

Each of the five outlets from the layer_indicator sub-patch, shown in figure 20, are connected to a list containing configurations for five gates that can either pass or block drum note data as shown in figure 21. For example if the first 16th note in the bar is the current position, rhythmic layer 5 is activated and the list of gate configurations associated with that layer will be selected and the corresponding 'g' values will be sent to open or close the *spigot* objects. This closes all of the gates associated with other layers 1 though 4 and sends a '1' to the logical AND gate connected to the right inlet of the *spigot* object. The other control line connected to the AND gate is derived from the user input. If both control lines are non-zero then the gate associated with layer 5 will pass any drum note data present at its input.

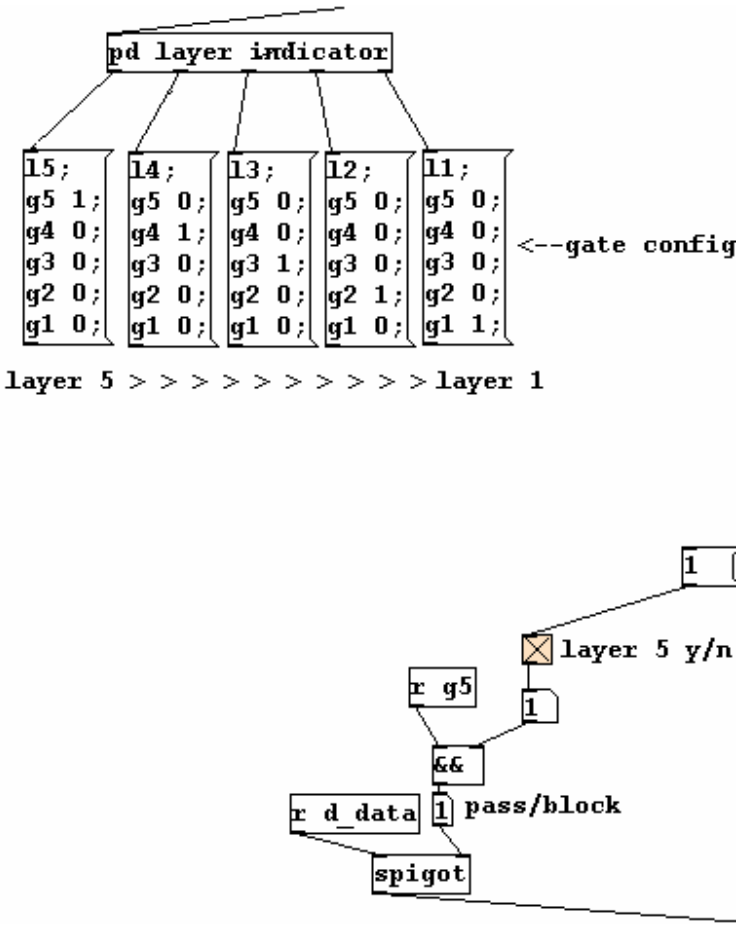


Figure 21: Layer indicator connected to gate configuration lists to control the operation of the *spigot* object

Incoming drum note data is split into groups of percussion instruments as follows: Bass Drums, Snares and Rim Shots, Hi-Hats and Ride Cymbals, Tom-Toms, Other Cymbals, Other Percussion. Each percussion group can be passed or blocked according to user input. This functionality is provided by the `bypass_rhythmic_variations` sub-patch.

Beat layers 1 and 2 can be toggled on or off according to the control values used in the mapping stage. It is possible to implement the switching of beat layers 3 and upwards but this was not implemented in the final version. This decision was taken to make the differences in the percussion track around the Activity-Valence control space, sound more musically natural. If the percussion rhythm was manipulated by toggling all of the five beat layers on and off, the measure of the score was found to be quickly lost and sound very disjointed. Positions with higher activity levels contain more percussion instrument groups and parts to give the impression of higher energy levels.

No rhythmic changes are imposed on the melody line.

9 Manipulation of Mode

9.1 mode_manipulator Sub-patch

The mode of the composition can be changed from major to either the parallel harmonic or Aeolean minor. All pitched note information is sent to the `mode_manipulator` where it is analysed and altered if necessary to produce the current selected mode. An algorithm calculates the scale degree that the current note lies upon. This can be described as follows:

```
if (current note value > 12)
{subtract 12 until result is < 12}
else pass result

subtract No.of semitones key tonic lies above C from result

if (result < 0)
{add twelve to result}
else pass result

result = final result
```

Tables 4 and 5 show the alterations that need to be performed to move from a major mode to either the parallel harmonic or Aeolian minor. If the current scale degree matches a degree specified to be altered in order to transform mode, then a transform takes place. For example if the first degree of a scale is represented by zero, then the fourth degree would represent an interval four semitones above, a major 3rd. This note would be flattened by a semitone to produce the correct pitch for that degree of the harmonic minor scale. To apply a flattening of one semitone to the current note a value of 1 is subtracted from its MIDI pitch value and the result sent to the outlet of the `mode_manipulator`.

10 Timbre & Instrumentation

The General MIDI (GM) standard, listed in Appendix D, defines a list of 128 voices (instruments) that can be selected using program change messages. To carry out a program change the *pgmout* object can be used. The left inlet should receive the voice number to change to and the right inlet should receive the number of the MIDI channel number to address.

Figure 7 states the happy area of the Activity-Valence control space should contain sharp harmonics, the tender and sad areas soft harmonics, while there are no specified harmonic qualities for the angry area. The specification of sharp or soft harmonics is as detailed as research to date goes with regard to how the timbre of instruments affects emotional expression (Jusiln, 2001). The following section describes how the instrument voice for the melody line is changed within the Activity-Valence control space. The instruments used are an interpretation of what could be suitable for mapping and not based on any significant findings, but on the Author's perceived timbral qualities.

10.1 instrumentation Sub-patch

The voice for the melody part is selected according to position of the cursor in the user input space. The space is divided into five sections as illustrated in figure 22. The sub-patches `happy_vol`, `sad_vol`, `angry_vol` and `tender_vol` all function as follows: Cartesian (x,y) co-ordinates are received and tested to see if the user cursor is within the specific area defined for each corner emotion of the Activity-Valence space. This is achieved by defining a square area using the *less than/greater than* objects. A positive number is sent to the outlet of these objects if the input condition is met. If both of the outlets have positive numbers present then the cursor is in the region defined. This check is carried out by a logical AND gate. For example in the `sad_vol` sub-patch shown in figure 23, if the x co-ordinate is less than -0.4 AND the y co-ordinate is less than -0.4 then send a '1' to the output of the gate.

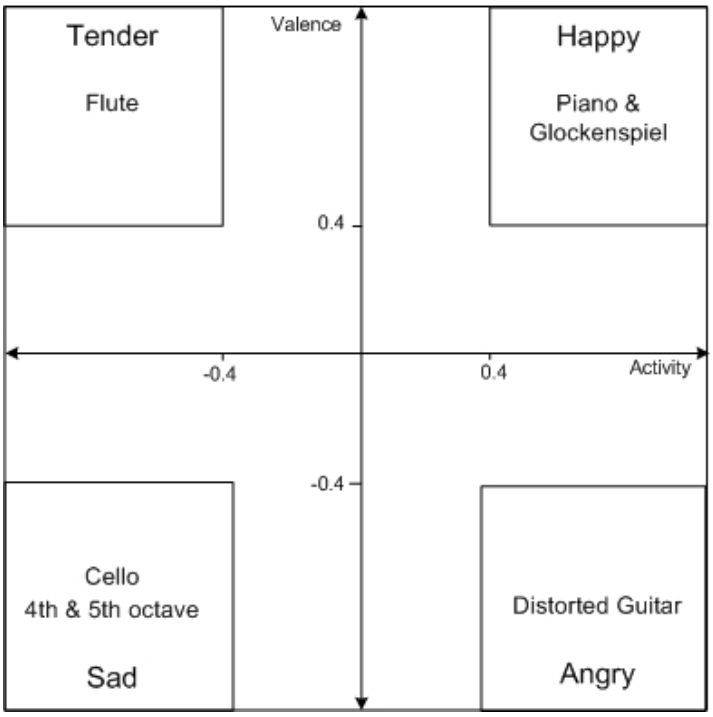


Figure22: Activity-Valence space with instrumentation areas superimposed

Transitions between voices are smooth, achieved by cross-fading the current voice and channel with the next voice and channel. The melody has five MIDI channels assigned to it. A counter is used to fade the MIDI volume of the each melody volume track up or down between zero and 100 as needed. A metronome running at a clock speed of one bang every 20ms is triggered every time there is a change in the status of the AND gate. If the change is from '0' to '1', the *spigot* object that implements an incrementing counter is enabled to fade up the MIDI volume for the specific MIDI channel. When a counter value of 100 is reached, the stop method is called and used to stop the *metronome* counting. The reverse process of a fade down takes place if the change in the status of the AND gate is from '1' to '0'.

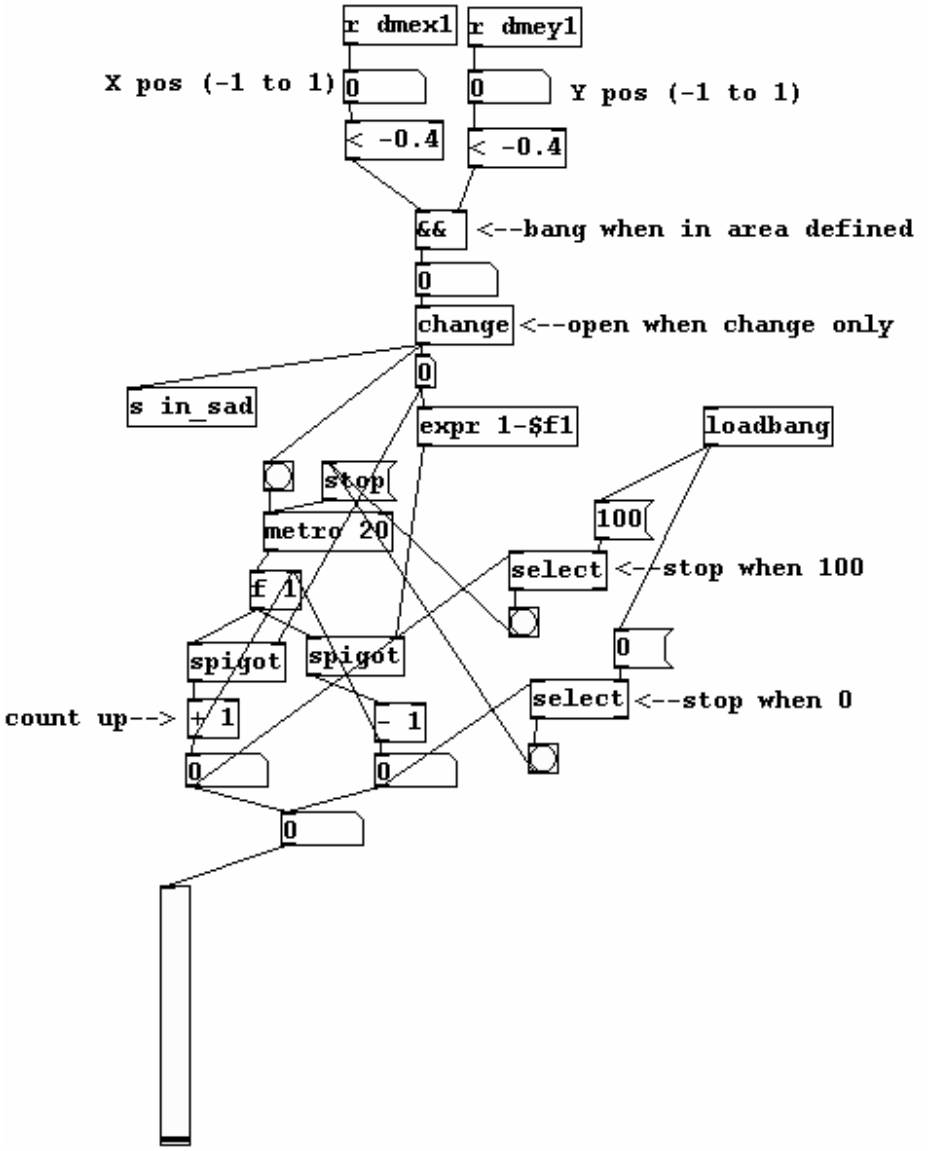


Figure 23: `sad_vol` sub-patch illustrating the use of the *metro* object to control MIDI channel volume

Transitions between voices are smooth, achieved by cross-fading the current voice and channel with the next voice and channel. The melody has five MIDI channels assigned to it. A counter is used to fade the MIDI volume of the each melody volume track up or down between zero and 100 as needed. A metronome running at a clock speed of one bang every 20ms is triggered every time there is a change in the status of the AND gate. If the change is from '0' to '1', the *spigot* object that implements an incrementing counter is enabled to fade up the MIDI volume for the specific MIDI channel. When a counter value of 100 is reached, the stop method is called and used to stop the *metronome* counting. The reverse process of a fade down takes place if the change in the status of the AND gate is from '1' to '0'.

The `original_vol` sub-patch functions in a similar way to the other `*_vol` sub-patches described. The difference in this sub-patch is that it calculates if the cursor is in any of the corner emotion areas using three logical OR gates and then triggers the counter when the status of the OR gate changes. The output of the counter is inverted by subtracting 100 and multiplying by -1 to make the result increment if the transition of the OR gate is from ´1´ to '0'. The counter also decrements if there is a '0' to '1' transition at the output of the OR gate. The MIDI channel volume for the original voice melody therefore is faded down to zero if the user cursor enters any of the corner areas specified, and faded up if any of the areas are not occupied.

To implement the MIDI volume changes for each channel, the values from each counter are sent to the instrumentation sub-patch from each of the *_vol sub-patches. These values are used to create a MIDI control message number 7 (volume) for each channel. The main melody, as specified in the score format, is always on MIDI channel number 1. To create a MIDI control message, a *ctlout* object is used. This object receives the counter value at its leftmost inlet, the message number (7) at the middle inlet, and the MIDI channel to address at the rightmost inlet. For the main melody part, the original voice is on channel 1, the angry voice on channel 11, the happy voice on channel 12, the tender voice on channel 13 and the tender voice on channel 14. Instruments for each voice are shown in figure 16. Each of the channels described extra to the main melody channel (MIDI channel 1) need to receive all the pitch, and duration data as channel 1. This is implemented using a *trigger* method for pitch and for duration to split the incoming melody note data into five identical copies. Each copy of the data is then combined with each of the channels used, using five instances of the *pack* method, and then sent to the five leftmost outlets of the instrumentation sub-patch. In effect, there are always five instruments playing the melody line, but only one of the channels has its' MIDI volume raised to 100. A log scale is used for the faders that control the MIDI volume to create a smoother cross-fade than a linear scale produces.

10.2 Doubling up of Melody Line

The melody line is copied and sounded transposed up an octave in the happy emotion corner and transposed an octave down in the sad emotion corner. This parallel melody line occurs on MIDI channel 15 and is implemented using the same method of fading the MIDI volume up and down in the `double_vol` sub-patch as described above. A program change to voice number 11 (glockenspiel) is applied and 12 semitones added to the melody pitch when the cursor enters the happy emotion corner triggered by the happy_vol sub-patch sending a bang labelled 'in_happy'. Glockenspiel was chosen for its bright and positive sound. A cello' was chosen for its slightly dull timbre at a lower register. A delay of 25ms is also added using a *pipe* object to emphasise the second part. 12 semitones are subtracted when the user cursor enters the sad emotion corner and a program change to voice 15 (cello) is applied by receiving a bang form the *receive* object labelled 'in_sad'.

10.3 drum_instrumentation sub-patch

The percussion part can also have changes in instrument. These changes are restricted to pitch changes on MIDI channel 10. A complete list of the GM1 Percussion sound set is listed in Appendix D. Incoming drum note data is split into groups of percussion instruments as described in the `layer_indicator` sub-patch by routing outputs of a *select* object to gates to pass or block the data for each percussion group. A rim-shot can be changed to a snare drum and a closed hi-hat can be changed to an open hi-hat depending upon the state of the control value derived from the user input (*rim2sn* receive object). Both changes produce a percussion track with more energy and drive. When enabled using the *spigot* gate object, the rim-shot to snare drum change is provided by using a *select* object. This object receives the pitches of the snare drum percussion group, 37, 38 or 40 (rim-shot, acoustic snare or electric snare respectively), and routes all rim-shots and acoustic snares to one output to trigger a message box with the number 38 stored inside (a snare drum).

The changes in Timbre are limited to changes in synthesised instruments from the GM standard. There could be smoother changes if a synthesiser, using FM synthesis for example, were used to provide the melody line. Many parameters can be changed to alter the timbre of the sound produced. These parameters could be mapped to the Activity-Valence control space to produce smoother changes in timbre than by changing instrument. This feature has not been considered as an option for this project due to time restrictions.

11 User Input to Create Control Values

11.1 Activity-Valence Control Space

The user input consists of an Activity-Valence control space as used in pDM. This area has four quadrants with one basic emotion represented in each happy, tender, sad and angry respectively. Control values are defined for each corner, with one set for each basic emotion. These values are chosen based upon research findings in Section 2 and summarised in figure 7. These values are by no means the definitive and final choice for mapping, but were chosen to try and clearly represent each basic emotion. A list of the control values for each corner and emotion can be found in Table 8.

Rule	*Happy*	*Tender*	*Sad*	*Angry*
Mode	1.5	1.3	-1.25	-1
Harmony	1	-1	-1.3	-2.5
Rhythmic:				
Hi-Hats	-1	-1	-1	1
Rim Shot	1	1	1	-1
Bass Drum	1	-0.5	-0.5	1
Snare	1	1	-1	1
Cymbals	1	-1	-1	1
Hats/Ride	1	1	1	1
Toms	1	-1	-1	1
Other Perc.	1	1	-1	1
Rhythmic Layer1	1	1	-1	1
Rhythmic Layer 2	1	-0.5	1	1

Table 8: Suggested control values for each of the four corners of the Activity-Valence Space

Cartesian co-ordinates are read from the users' cursor position, and used to calculate intermediate control values. This calculation involves a linear interpolation (Friberg, 2004b) resulting in a skewed plane in the 2D space. The interpolation is carried out using *(2)* used in pDM (Friberg, 2004b):

$$k(x,y) = \frac{1}{4}\left[(k_h - k_t - k_a + k_s)x + k_h + k_t - k_a - k_s\right]y + \frac{1}{4}\left[(k_h - k_t + k_a - k_s)x + k_h + k_t + k_a + k_s\right]$$

(2)

Referring to *(2)*, x represents activity and y represents valence ranging from -1 to 1 and k_h, k_t, k_a, k_s ,represent the control values assigned to each corner, Happy, Tender, Sad and Angry respectively. Figure 24 illustrates the interpolation process for the harmony rule and the resulting control values.

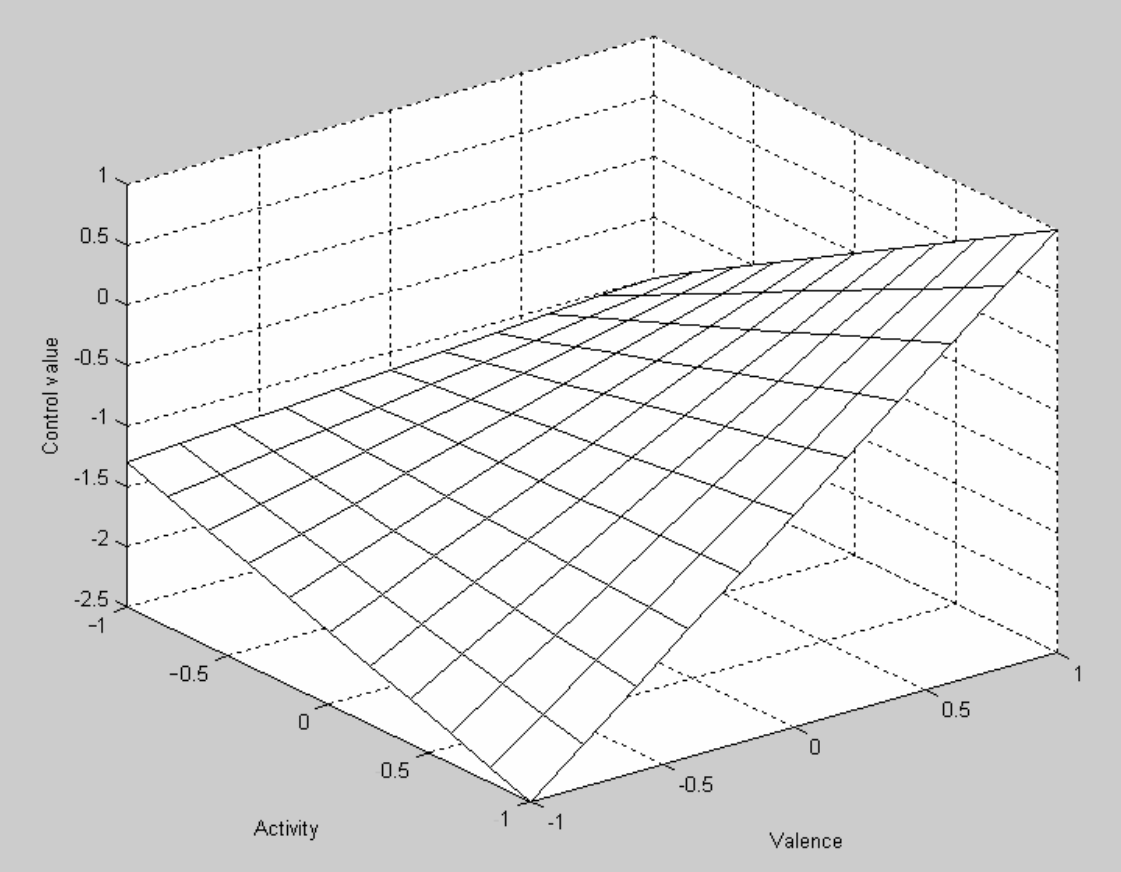


Figure 24: A graph showing the interpolation of control values for the harmony rule

11.1.1 `activity_valence_space_window` sub-patch

This window contains the values in Table 8 for each rule, and interpolates intermediate values using an implementation of equation1 in an *expr* object as shown in figure 25.

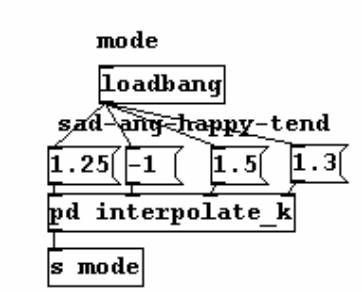


Figure 25: Control values stored in message boxes and sent to `pd` interpolate_k on start-up.

The sub-patch `activity_valence_space_window` is taken directly from pDM (Friberg, 2004b) and modified to produce the control values necessary for the manipulation of the structural factors of the input score. The `interpolate_k` sub-patch shown in figure 26, contains an *expr* object implementing *(2)*. The calculated control values are sent to the appropriate sub-patch using a *send* and *receive* object pair.

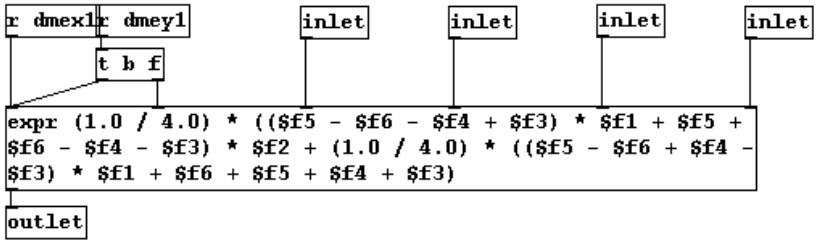


Figure 26: `interpolate_k` sub-patch showing *expr* object containing implementation of *(2)*.

The `activity_valence_space_window` sub-patch also contains the `pdm_mouse_input sub-patch` that provides the facility to open a GEM window that is the Activity-Valence control space. GEM is the Graphics Environment for Multimedia. GEM is a collection of externals which allows the creation of real-time graphics for `pd`. The four basic emotions are labelled, one in each corner, and mouse movements inside the window are tracked producing Cartesian (x,y) co-ordinates ranging from -1 to +1.

11.1.2 Mode Mapping

Initially, the mode mapping was set so that the two uppermost corners of the Activity-Valence control space were assigned a value of 1, and the two lowermost corners were assigned a value of -1. After interpolation, this resulted in a value that changes sign when the user crossed the x-axis. This change in sign is the point at which the mode changes from major to minor or vice versa. The mapping as it stands now, is based around the results of Hevner shown in table 1 and is given in table 8. Hevner places a heavier weighting on a major mode for a happy emotional expression than for a gentle emotional

expression. There is also a heavier relative weighting for a sad emotional expression. The values chosen reflect Hevner's weighing and produce a skewed plane as illustrated in figure 27.

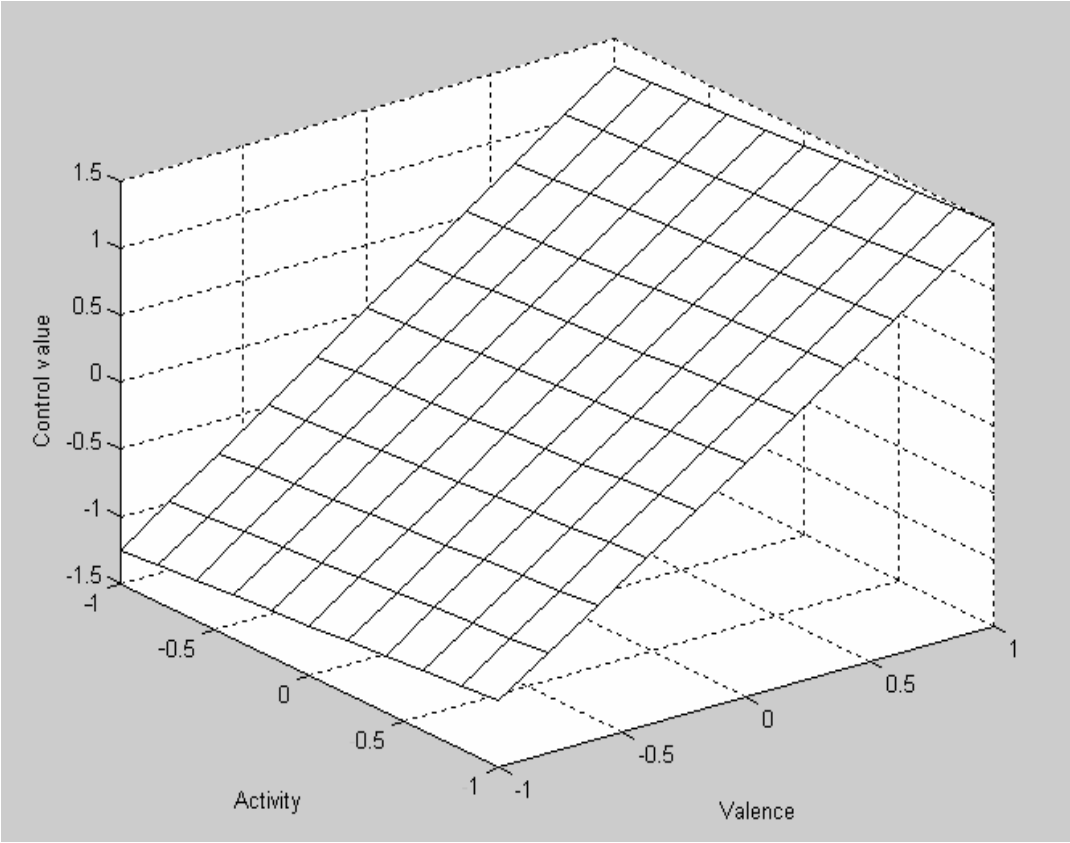


Figure 27: A graph showing the interpolation of control values for the mode rule

11.1.3 Harmony Mapping

Harmony control values greater than 0.2 will drop any tensions specified in the score and play either major or minor triads. Values less than or equal to 0.2 will play all tensions as specified in the score.

With regard to the voicing of the chords, positive harmony control values will cause all chords to be played in root position. Control values less than zero and greater than or equal to -1 will play chords with voicing as illustrated in figure 13(a). Control values less than -1 and greater than or equal to -1.25 will play chords with voicing as shown in figure 13(c). Control values less than -1.26 and greater than or equal to -2.5 will play chords with voicing as shown in figure 13(d) in the lowest register. An additional dissonant fourth is added when the control value is less than -1.75. Values for each corner of the Activity-Valance control space have been chosen to reflect the properties shown in figure 7 that relate to the accompaniment such as harmony complexity and melodic range. Figure 24 shows the skewed plane created by interpolating the harmony control values throughout the Activity-Valence control space.

11.1.4 Rhythmic Mapping

For the *Hi-Hats* a positive control value changes the instrument from a closed hi-hat to an open hi-hat. In similar fashion, a negative value for the *Rim Shot* control value causes a rim shot to change to a snare drum. Both of these changes are implemented in the `drum_instrumentation` sub-patch.

The remaining rhythmic control values in table 8 mute or un-mute the named instruments for negative and positive values respectively.

11.1.5 Rhythmic Layer Mapping

Rhythmic layers 1 and 2, as described in section 4.2, are blocked with negative control values from these two mappings. Layers 3 and upwards are left un-blocked at all times to keep enough rhythmic consistency in the score.

11.2 Main Window

The main window provides the user controls to open, play, stop and loop a score. The current bar number is also shown as received from the bar object in the input score. Three sub-patches appear in the window. Sequencer sub-patch provides the main functionality of the whole patch as illustrated in figure 28. The `rule_control_window` sub-patch contains the user controls to alter the type of structural alterations to apply to a score. The `activity_valence_mapping_window` is described above.

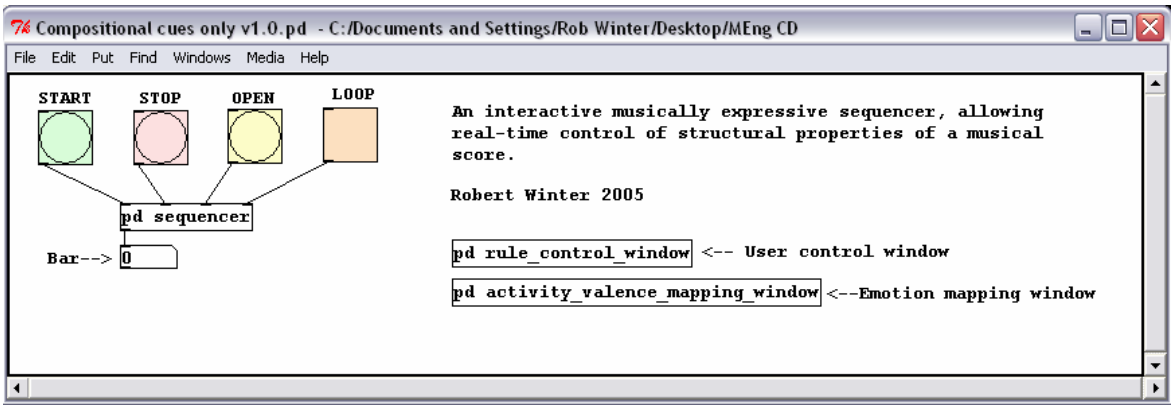


Figure 28: Main Window

11.3 Rule Control Window

The rule control window, shown below in figure 29, in default start-up status, enables the user to bypass any particular structural change applied by the application by checking the any of the top four toggle boxes. Each of the toggle boxes provide a control value named either `bypass_instrumentation`, `bypass_rhythmic_variations` and `bypass_harmony_variations` to enable or disable the sub-patches with the same name as the control value.

Any combination of the four voices used, bass, melody, percussion and accompaniment, can be muted by checking the appropriate toggle box. This can be useful to hear each of the structural rules applies individually. In a similar fashion to the control values for the

bypass rule toggle boxes above, the mute toggle boxes send control values to the `mute_control` sub-patch.

The final toggle box allows the minor mode applied to be switched from Aeolian minor (default) to Harmonic minor. This toggle box sends the control value named 'mode type' to the `mode_manipulation` sub-patch to switch between the *select* objects that process incoming note data to apply mode changes.

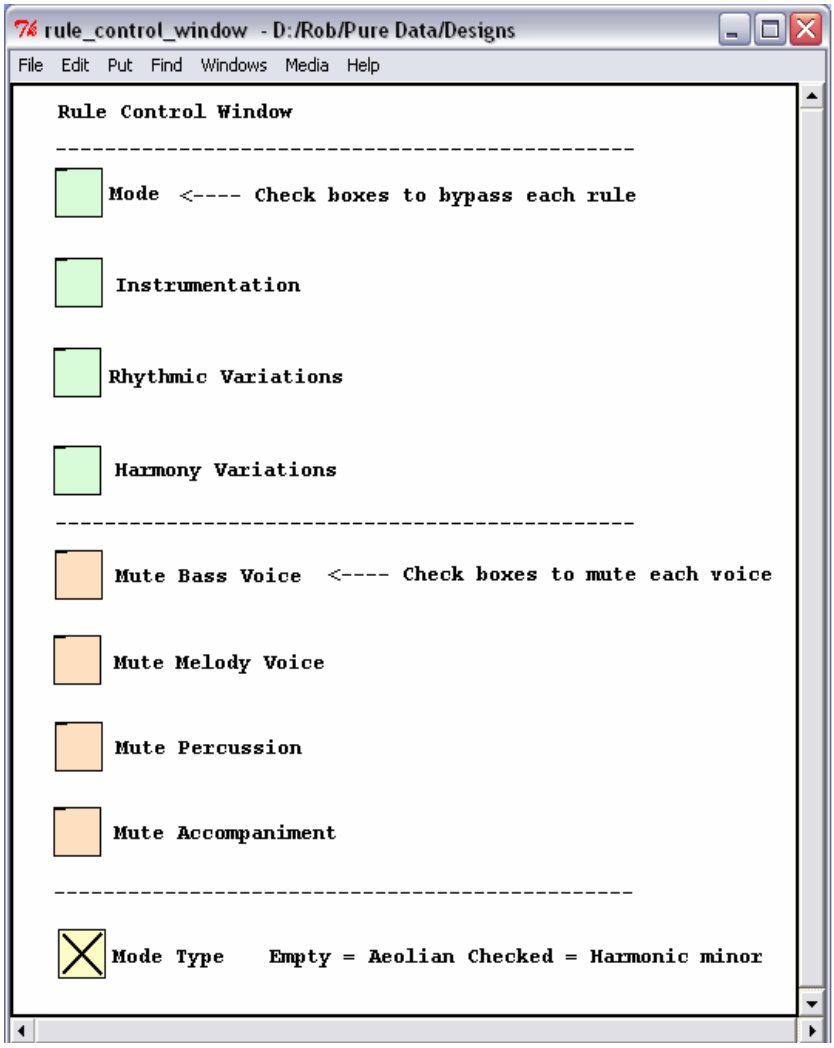


Figure 29: Rule control window

11.4 Sub-Patch Hierarchy Diagram

Figure 30 shows the hierarchical structure of the sub-patches in the full system to illustrate which sub-patches are contained within others.

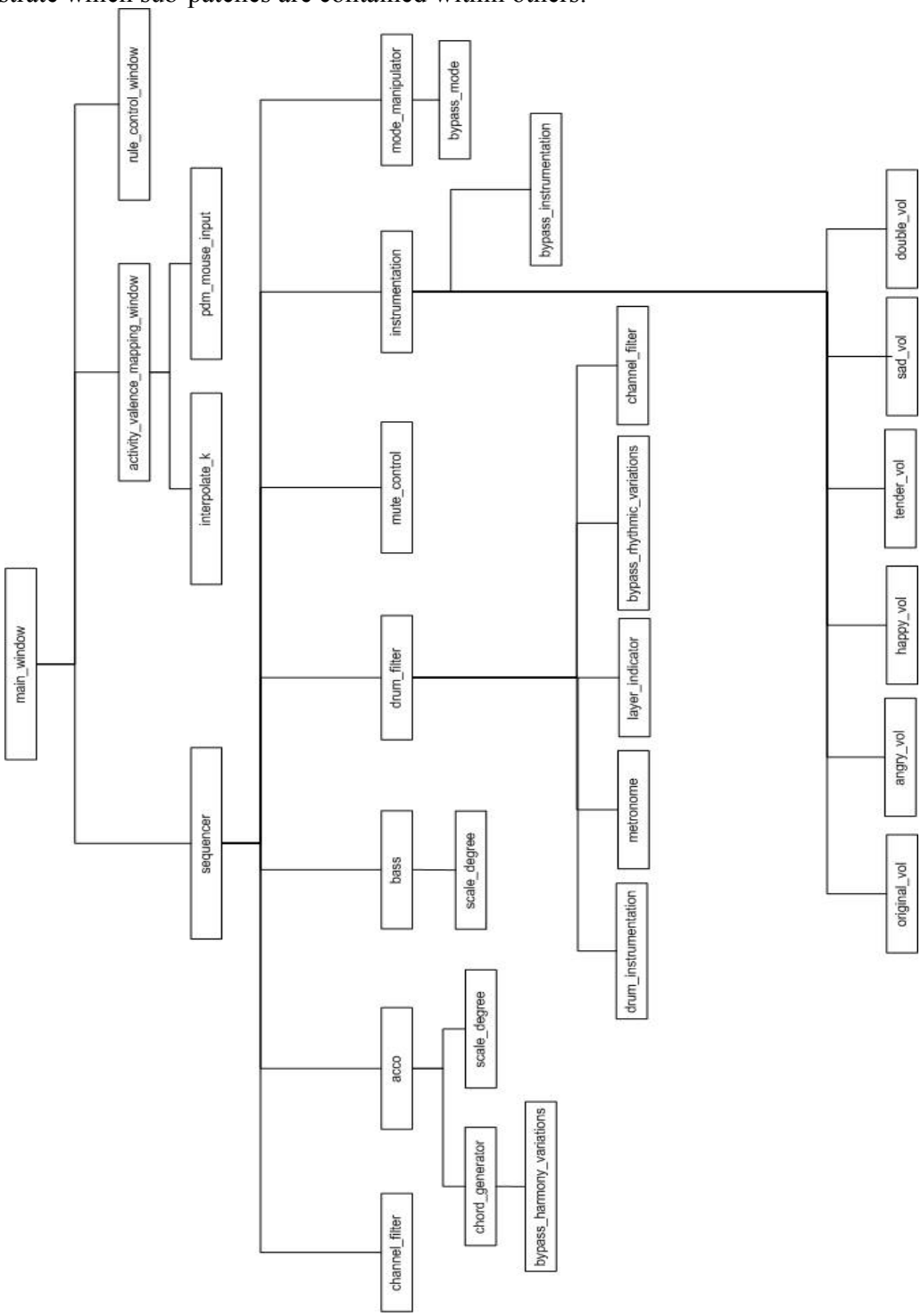


Figure 30: A diagram to show hierarchy of sub-patches in the system

12 Integration with pDM

Integration of the system with pDM to produce a system that allows the real-time control of the performance and compositional cues together has been implemented. Acoustic cues such as volume, timing and phrasing are added to build a complete system that can control performance and compositional factors together.

The program has been designed to allow for integration with pDM. Pure Data was chosen as the development environment as this is the same used for pDM. The score format used is an extension to the format used in pDM. This is fully compatible with pDM. Figure 31 shows how the system is integrated with pDM.

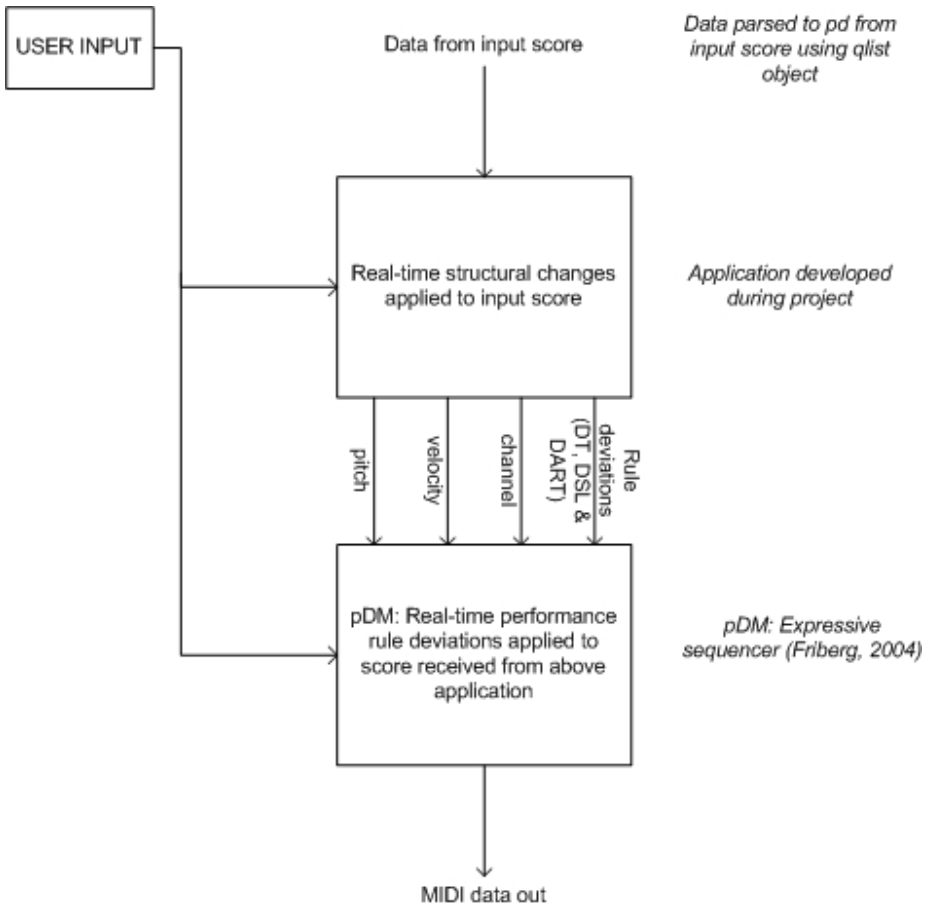


Figure 31: Block diagram showing how the application developed during this project is integrated with pDM.

The screen shots in Appendix C from *(cc)* onwards show how there is one *qlist* sequencer object used for both the application developed during this project and pDM. This is also the case for the user controls. Mapping for the control values in the Activity-Valence control space is merged with the `pdm_activity_valence_mapping_window` sub-patch. The `rule_control_window` sub_patch is contained in the

`pureDM_expanded` window to allow user control over the type of compositional alterations that are applied.

13 System Testing

System testing was carried out during the implementation stage of the project. Each sub-patch was tested in isolation to ensure that it was operating as required before integration. This was mainly carried out using a MIDI keyboard as a live input to the system to replicate the note data in the input score. The output was monitored to verify operation. A discussion of the performance of the system related against the original objectives and acceptance testing is presented below to check if the system has delivered what was required of it. Examples of the system output with the user position at each of the corner positions in the Activity-Valence control space can be found on the accompanying CD-ROM.

13.1 Choice of Test Scores

Two classic popular songs were chosen, A Hard Days Night and Here There and Everywhere, both written by The Beatles. The first choice represents a simple pop song that is a synthesis of blues and rock harmony using the I-IV-V three-chord changes (G, C, and D, respectively) in the standard 12-bar blues form plus a I-bVII-I (G-F-G) cadence. The second is a more down-tempo song with a more complicated chord progression that modulates key towards the end of the piece. The different chord tensions present in the score also demonstrate the functionality of the accompaniment generator. Notated musical scores of the two test scores illustrating the melody line and chord progression can be found in Appendix I.

13.2 Fulfilment of Objectives and User Acceptance Testing

Tests need to be carried out to find out if the objectives specified in section 3 have been met by the system. Once this process is complete, user acceptance testing (UAT) can then be carried out. The objective of UAT is to get real users to try to make the system fail, in the context to which the system is designed for use. This context can be defined by the project objectives. To re-iterate the project objectives can be stated again from section 3:

1) Structural changes shall be applied to a pre-composed score in real-time, and be controlled by the user input.

2) User input will take the form of the Activity Valence control space.

3) Musical output shall demonstrate the structural qualities defined in each corner of figure 7 when the user input position is located at co-ordinates (-1,1), (1,1), (-1,-1) and (1,-1).

4) The input score shall contain a melody line with a specified harmonic progression, and a percussion part.

5) An accompaniment and bass part shall be generated using the harmonic progression specified in the score as a guide. These parts shall reflect the structural qualities defined in figure 7 with regard to the mode, harmony and pitch variations.

6) Structural changes to the musical output shall be applied in a gradual manner between the basic emotions when user input position is changing.

7) The system shall be designed to allow for and to be integrated with pDM to create a real-time musically expressive sequencer that models the performer along with the composer.

From observation, objectives 1 and 2 have been met as a pre-composed score is structurally altered in real-time, controlled by a user by means of an Activity-Valence control space. With regard to objective 3, some of the specific structural properties at the co-ordinates defined, such as mode and consonance, have been met. Interpretations of the definitions for rhythm have been made. For high positive activity values, the properties of the rhythm component were defined as being either irregular or both irregular and regular. These definitions have been refined for this system so that the rhythms for the percussion track are related to energy rather than regularity as discussed in section 8.3. For soft and sharp harmonics, a suggested instrument voicing for each emotion is implemented, but as discussed in section 10, this is not based on any established research on timbre and emotional expression. The accompaniment part demonstrates large or small pitch variations as specified in the happy and sad emotion areas respectively. Figures 32 and 33 below show how the accompaniment part differs for the same chord change when the user is at positions (1,1) and (1,-1), happy and sad respectively. The accompaniment for the chord progression from C Major to D Major for the happy emotion area has a large pitch leap from the original chord, whereas the second the two chords in the sad emotion area are voiced very close together resulting in a small pitch change. Figure 33 includes the transformations applied by the change from major to minor mode and an additional dissonant fourth.

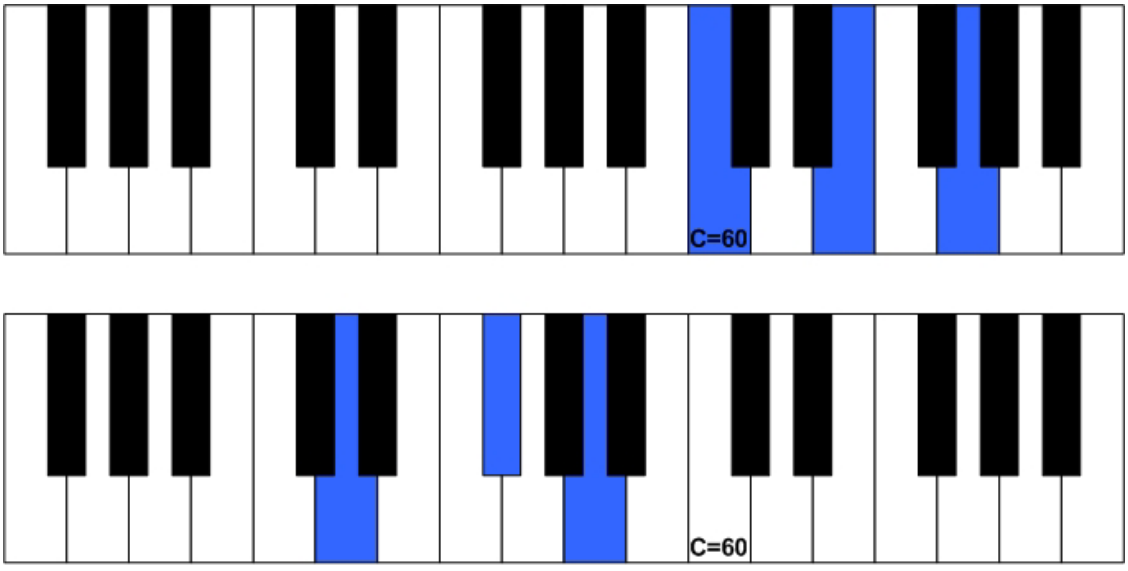


Figure32: Accompaniment part when user is in position (1,1), happy emotion area, showing pitch change when input chord changes from C Major 7 (top figure) to D Major 7 (bottom figure).

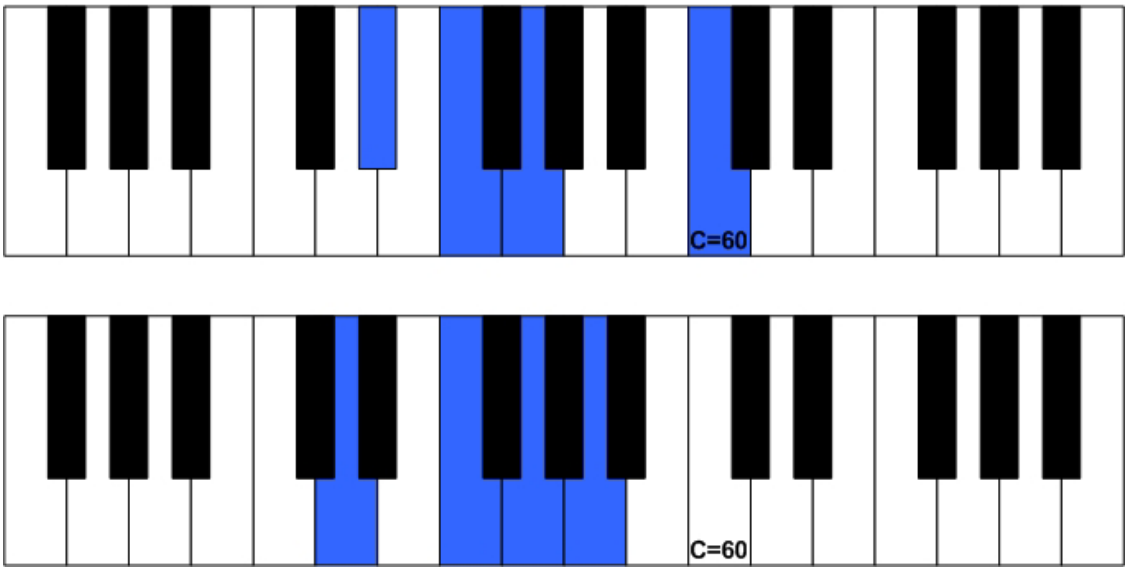


Figure33: Accompaniment part when user is in position (1,-1), sad emotion area, showing pitch change when input chord changes from C Major 7(top figure) to D Major 7 (bottom figure).

A full record of the testing carried out is given in Appendix N. No change is made to the melody line except for changes in mode from major to minor.

Objective four stating that the input score should contain a melody line with a specified harmonic progression, and a percussion part is met. This format is adopted for the input scores as shown in Appendix G.

Objective five states that an accompaniment and bass part should be generated using the harmonic progression specified in the score as a guide. These parts should reflect the structural qualities defined in figure 7 with regard to the mode, harmony and pitch variations. The two parts follow use the chord progression specified in the score as a guide. The root note is always played as specified, however the tension of the chord is changed according to user input. The parts implement the structural qualities of each corner emotion with regard to mode, harmony and pitch variation.

Objective six states that structural changes to the musical output should be applied in a gradual manner between the basic emotions when the user input position is changing. Attempts to create smooth changes between the four specified corner emotions have been made by using control values interpolated within the Activity-Valence control space. The control values are used to define the amount and type of structural changes to apply to the input score. With the harmony rule, there are four corner states at each extreme and two intermediate states that the accompaniment can take. As the values on the valence axis approach larger negative values towards -1, the pitch register of the accompaniment is lowered in 3 steps. The percussion part changes in gradual steps with movement around the Activity-Valence control space. Beat layers 1 and 2 are toggled on and off to produce

changes in rhythm along with the removal/addition of selected percussive instrument groups. With changes in mode, there can only be a discrete major or minor mode selected as is also the case with instrumentation where there is a specific voice defined for each area. To overcome the problem sudden changes from structural factors that allow only two discrete states, the control values suggested map structural changes so that they change state at different points in the Activity Valence control space. This results in a build up towards the full structural properties defined in figure 7 for each emotion corner as the user approaches them.

Objective seven states that the system should be designed for and to be integrated with pDM. This has been achieved as the system was always designed to be integrated with pDM once complete. The two full systems were successfully integrated by the Author and A.Friberg, who developed pDM originally. Audio examples of the full system in operation are provided on the accompanying CD-ROM. See Appendix M for a full tracklisting.

UAT for this system is presented in the form of video examples of users running the system. These files can be found on the accompanying CD-ROM (see appendix M). During UAT and demonstrations of the system developed during this project to groups and individuals, comments have been made that there is a perceived tempo increase as values increase towards the left-hand side of the Activity axis, although the tempo is strictly fixed. Along with the perceived tempo increase, a perceived increase in volume was another common observation from test subjects. Again, the volume of the sequencer in the system developed, when not integrated with pDM, is fixed. These comments are encouraging as both of the perceived changes relate to an increase in activity or energy in proportion to the Activity axis of the Activity-Valence control space.

14 Conclusion

14.1 Summary

Using this application a new way of creating music in a standalone or multimedia application can be foreseen. A melody line along with a simple harmonic progression and percussion part can be transformed into a full piece of music with real-time control over the structural makeup of the score. It can be seen as a joining of the composer and listener. When combined with the performance rules, there is a fusing of the traditional composer with the performer.

Interface complexity can provide different levels of user control. The mapping of the so-called basic emotions, happy, sad, angry & tender, provides a high level simple user interface that can be understood by musicians and non-musicians alike. When acting at a composer level, control over specific structural compositional elements require a complex interface. When combined with performance deviations, the interface could becomes more complex with regard to the mapping, and may be unrewarding to control specific parameters. This leads to the possibility of having separate interfaces for control of composition and performance. A conducting baton for example could provide the interface for performance deviations and some other method to control the structural properties of a score. These could then be controlled individually by two separate users.

The question as to how effective the system will be at communicating the desired emotional expression using data collected from the literature survey remains to be answered. Listening tests could be carried out with subjects asked to describe the perceived emotional expression that they heard There are many different methods of conducting these tests including self-report feedback, interviews with subjects, brain imaging and questionnaire feedback. Such experiments lie outside the bounds of this project but it is worth considering some issues that may arise regarding the subjects' perception of emotional expression. The ecological validity of the test music using computer-synthesised instruments would be low in comparison to that of a real-life human performance. There are however, added benefits of using computer-synthesised instruments. Insight into the effect of specific cues can be gained by having direct control over the separation of them. Compositional cues can also be investigated in isolation without any performance cues influencing the perceived emotional expression. There is also the fact that emotions may be aroused by structural properties of a musical piece, whereas some sources of emotion in music reflect personal associations, for example nostalgia. This suggests there is an inherent personal and individual element in each listener's perception of emotional expression.

14.2 Conclusions

Using established research, a system has been created that implements real-time structural changes to a pre-composed musical score influencing the perceived emotional expression. Control over the type and amount of structural change is provided by an Activity-Valence control space. This space has four quadrants labelled as happy, tender, sad and angry. A set of suggested control values are assigned to each corner of the space

designed to represent each of the four basic emotions described. An interpolation of control values has proved effective to produce smooth transitions between corner emotions.

A concise model showing the nature of the key structural factors found to be significant to communicate specific emotional expressions has been created. This could also be combined with a 3D visual environment to provide more effective immersion in game-play and other multimedia environments. The system has been integrated with pDM to create a full system that allows the real-time control of both performance *and* compositional cues together.

15 Further Work

The following suggestions for further work and development can be made:

1. Expansion of the system developed for this project to handle scores with compound time signatures and swing grooves. Currently the rhythm manipulations are only able to function non-swing grooves. Scores written in a minor key are not able to be changed to a major mode at present. An extension of the system to bring this feature into the system would be a worthwhile improvement.

2. A program of listening tests to carry out an investigation as to the effectiveness of the application in communicating specified emotional qualities. Tests could be run using the application developed as part of this project without being integrated with pDM. The effectiveness of this application could then be investigated. Further listening tests could be run with the system integrated with pDM, to find out if the compositional techniques identified improve the reliability of the intended emotional expression.

3. Investigation into developing the structural changes further could be carried out. Developments to the simple mode changing algorithm could be made to avoid the problem of a chromatic scale changing to a diatonic scale with tone repetitions when the mode is changed to minor. If there is chromatic harmony present in the score there will also be tone repetitions. Research into structural changes that are based around probability, such as rhythmic changes based on Markov chains could produce more interesting, non-repeating, rhythmic alterations.

16 Acknowledgements

I wish to thank the following people for their help and support throughout the project duration:

Anders Friberg, Damian Murphy, David Howard, Andy Hunt and Sten Ternström.

Alastair Moore, John Winter, Jane Reid, Peter Winn, Dan Muscut and Hanna Essex.

17 Appendices

A. References

Déchelle, F. 1998. jMAX download site:
http://freesoftware.ircam.fr/rubrique.php3?id_rubrique=14

Friberg, A. 1995. "A Quantitive Rule System for Musical Performance" Stockholm: Royal Institute of Technology. Summary available at:
http://www.speech.kth.se/music/publications/thesisaf/sammfa2nd.htm
Accessed 06/05/05.

Friberg, A., and J. Sundberg. 1986. "A Lisp Environment for Creating and Applying Rules for Musical Performance" In *Proceedings of the International Computer Music Conference.* San Francisco: Computer Music Association.

Friberg, A. et al. 1991 "Performance Rules for Computer-Controlled Contempory Keyboard Music." *Computer Music Journal* 15(2): pp. 49-55.

Friberg, A. et al. 2000. "Generating Musical Performances with Director Musices" *Computer Music Journal*, 24(3), pp. 23-29.

Friberg, A. 2004a, Director Musices Program and Manual, available at:
http://www.speech.kth.se/music/performance/download/dm-download.html
Accessed 06/05/05

Friberg, A. 2004b, "pDM: An Expressive Sequencer with Real-time Control of the KTH Music Performance Rules" Department of Speech Music and Hearing, Royal Institute of Technology (KTH), Stockholm.

Gabrielsson, A. and E. Lindström . 2001, "The influence of Musical Structure on Emotional Expression" *Music and Emotion: theory and research*, Oxford University Press. pp223-248.

Hobbis, J. 2003, "Interactive Music for Computer Games" *4th Year Project Report for Degree of MEng in Electrical Engineering with Music Technology Systems*, University of York, York, United Kingdom.

Howard ,D and Angus, J. 1998, *Acoustics and Psychoacoustics*, Focal Press. pp74-79 & pp138-144.

Juslin, P.N., 2001, "Communicating Emotion in Music Performance: A Review and Theoretical Framework" *Music and Emotion: theory and research*, Oxford University Press. pp309-337.

Juslin, P.N & Linström, E. 2003. *"Musical expression of emotions: Modelling composed and performed features.* Manuscript submitted for publication.

Jusiln, P.N & Laukka, P, 2004 "Expression, Perception, and Induction of Musical Emotions: A Review and a Questionnaire Study of Everyday Listening" *Journal of New Music Research*, 33(3), pp 217-238.

Lindström, E. 1997. "Impact of Melodic Structure on Emotional Expression" In *Proceedings of the Third Triennial ESCOM Conference, Uppsala, June 1997, (ed. A. Gabrielsson),* pp. 292-7. Uppsala, Sweden: Uppsala University.

McAlpine. K, Miranda. E, Hoggar. S. 1999 "Making Music with Algorithms: A Case-Study System" *Computer Music Journal*, 23(2), pp19-30.

Microsoft Corporation, 2005. DirectX 9.0, available at:
http://www.microsoft.com/downloads/search.aspx?displaylang=en&categoryid=2
Accessed on 18/02/05

Microsoft Corporation, 2003. DirectMusic Producer download:
http://www.microsoft.com/downloads/details.aspx?FamilyID=6e938a6e-b383-466b-a3ee-5a655bf5db8c&displaylang=en
Accessed on 18/02/05

Moore, B.C.J & Glasberg, B.P. 1983. "Suggested Formulae for Calculating Auditory Filter Bandwidth and Excitation Patterns" *Journal of Acoustic Society of America*, 74(3), pp 750-753

O'Donnell, M, 2002. 'Producing Audio for Halo'
http://halo.bungie.org/misc/gdc.2002.music/
Accesed 05/06/05

Puckette, M 1989. MAX/MSP download site:
http://www.cycling74.com/products/dlmaxmsp.html
Accessed 05/06/05

Puckette, M. 1996. "Pure Data" *Proceedings of the 1996 International Computer Music Conference.* San Francisco: International Computer Music Association, pp 269-272.

Scharf, B. 1970. "Critical Bands" *Foundations of Modern Auditory Theory,* Vol. 1, London: Academic Press, pp 159-202.

Temperley, D & Sleator, D. 1999. "Modelling Meter and Harmony: A Preference Rule Approach" *Computer Music Journal*, 23(1), pp10-27

Temperley, D. 2004a. "An Evaluation System for Metrical Models" *Computer Music Journal,* 28(3), pp28-44.

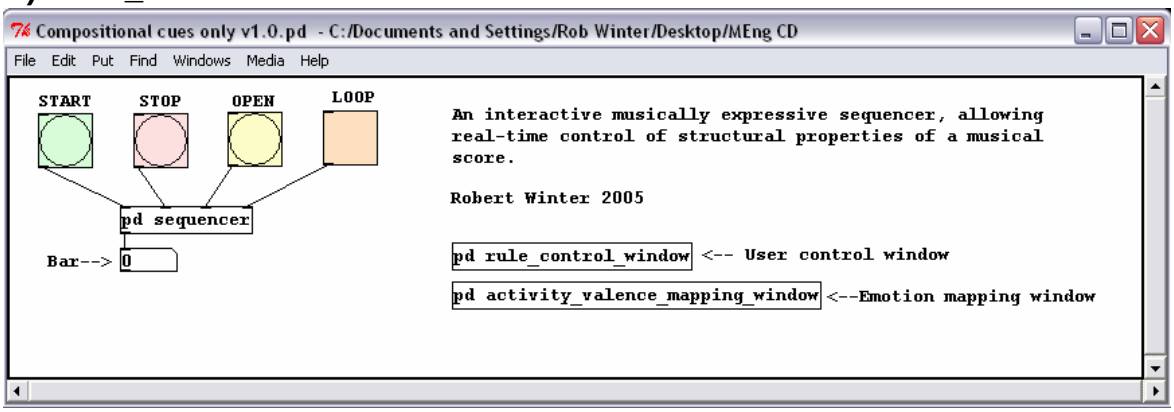
Temperley, D. 2004b. Ftp site: ftp://ftp.cs.cmu.edu/usr/ftp/usr/sleator/melisma2003/
Accessed 18/02/05

B. Bibliography

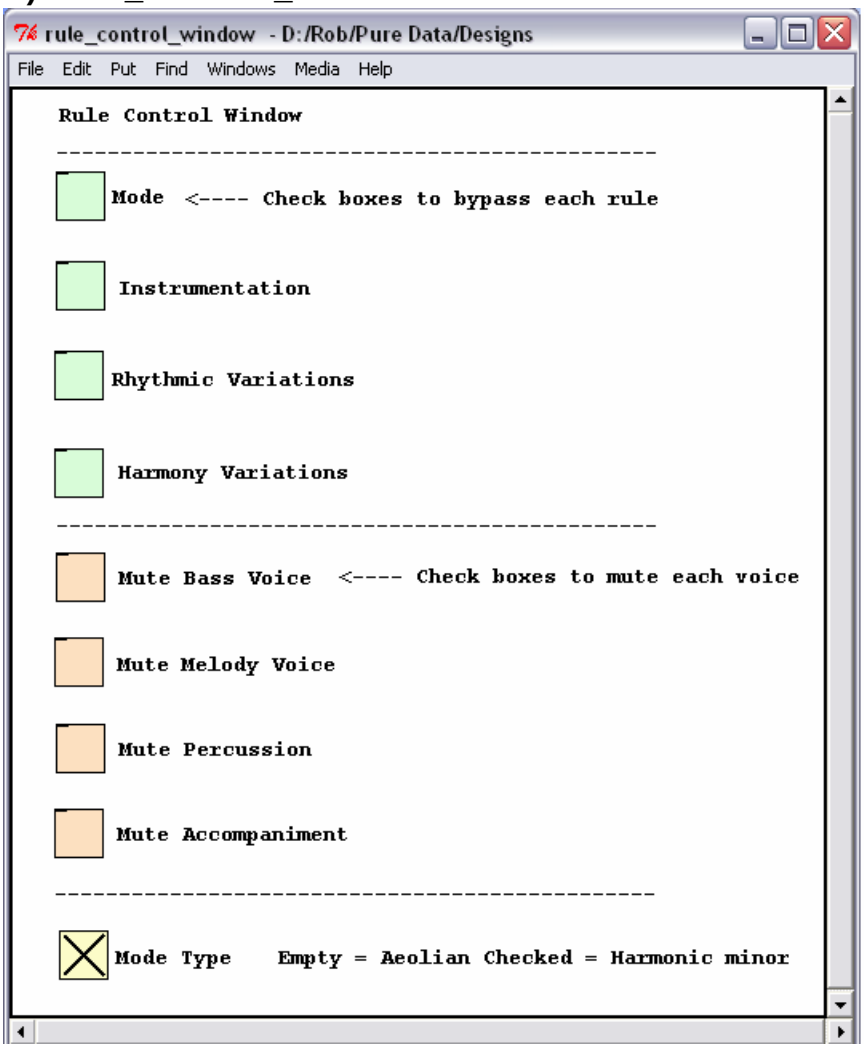
Aikin, J. 2004, *A Player's Guide to Chords and Harmony*, Backbeat Music Essentials Series, Backbeat Books.

C. Pure Data Patches

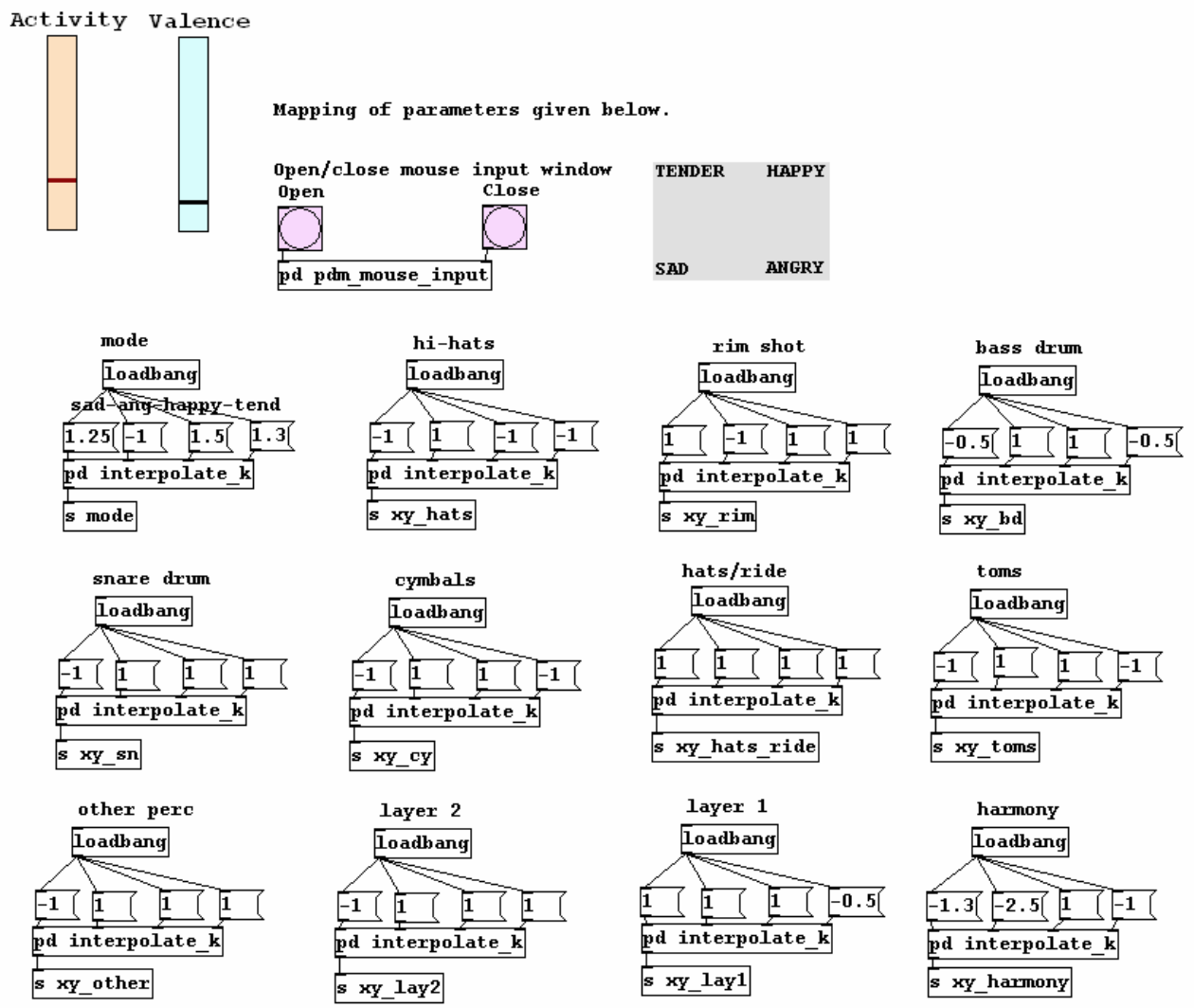
a) main_window



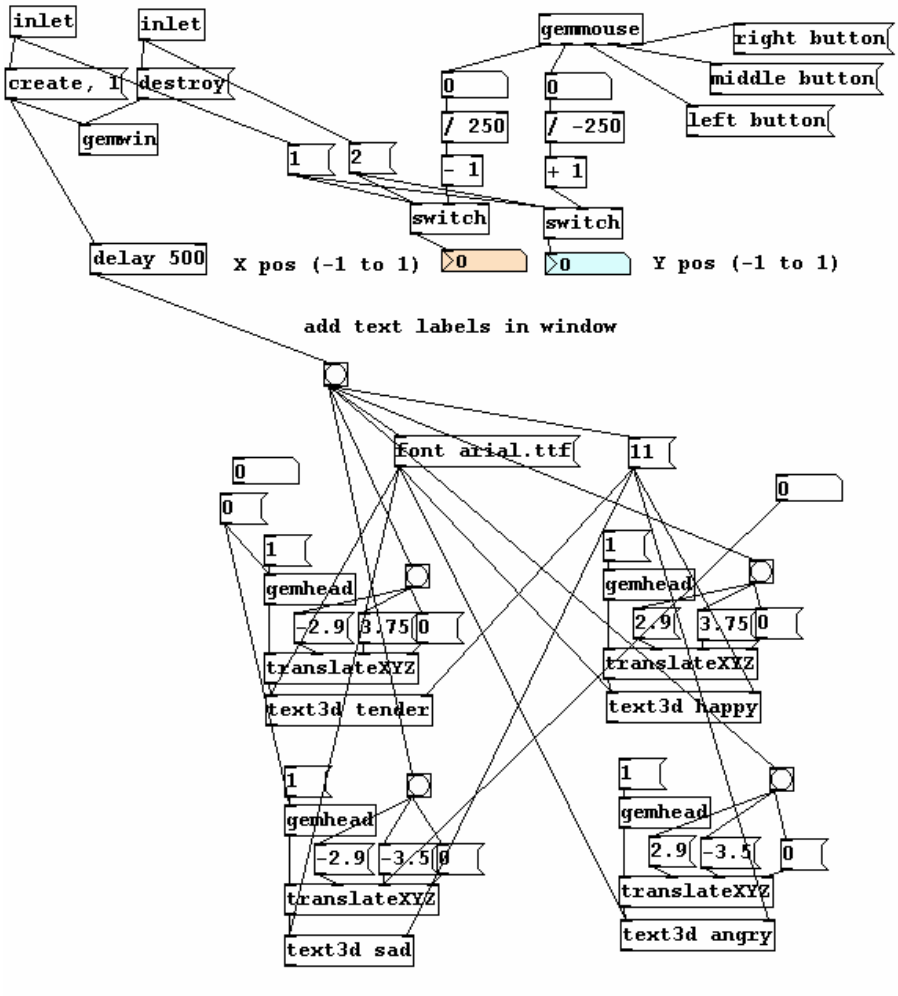
b) rule_control_window



c) activity_valence_mapping_window

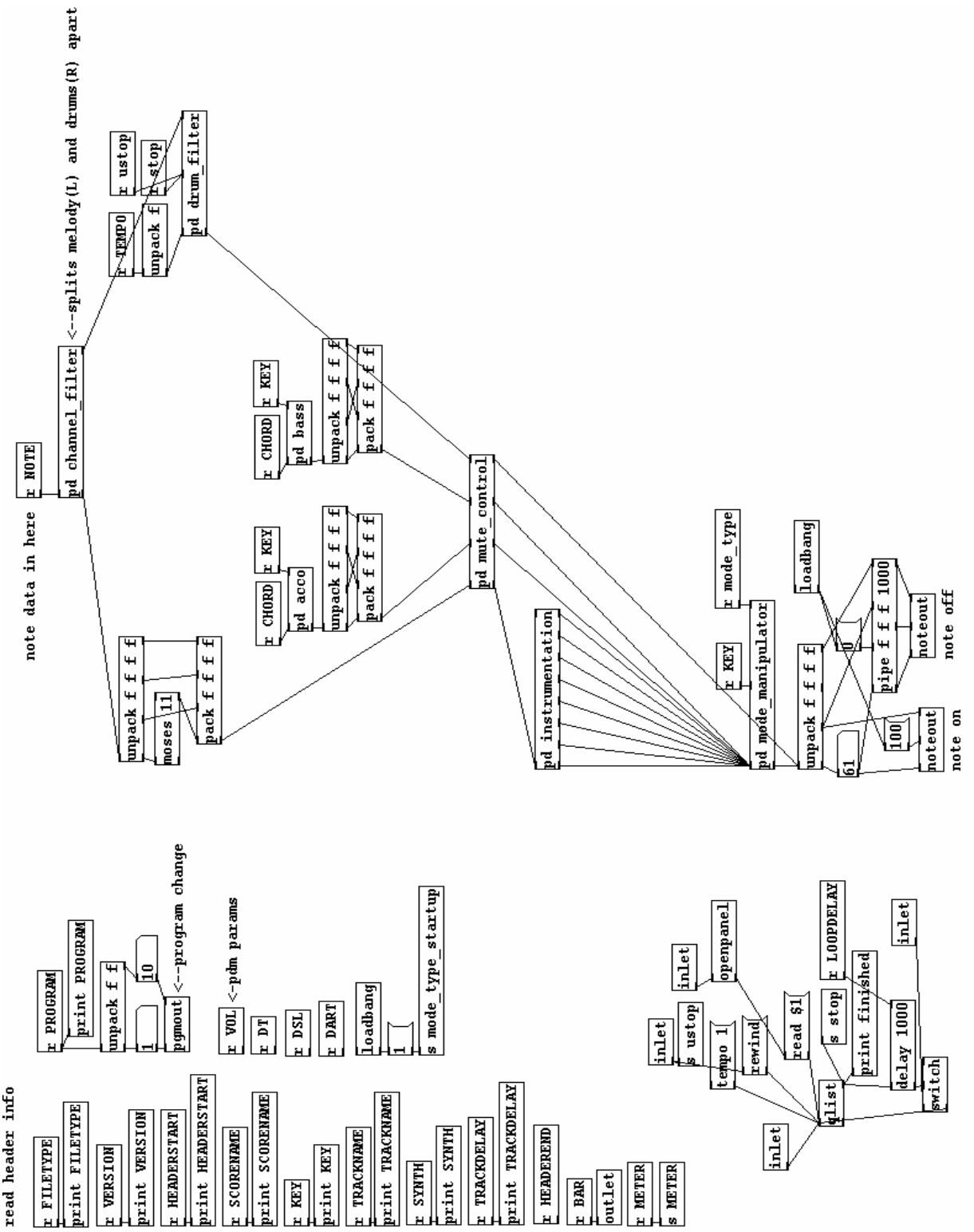


d) `pdm_mouse_input` (Friberg, 2004)

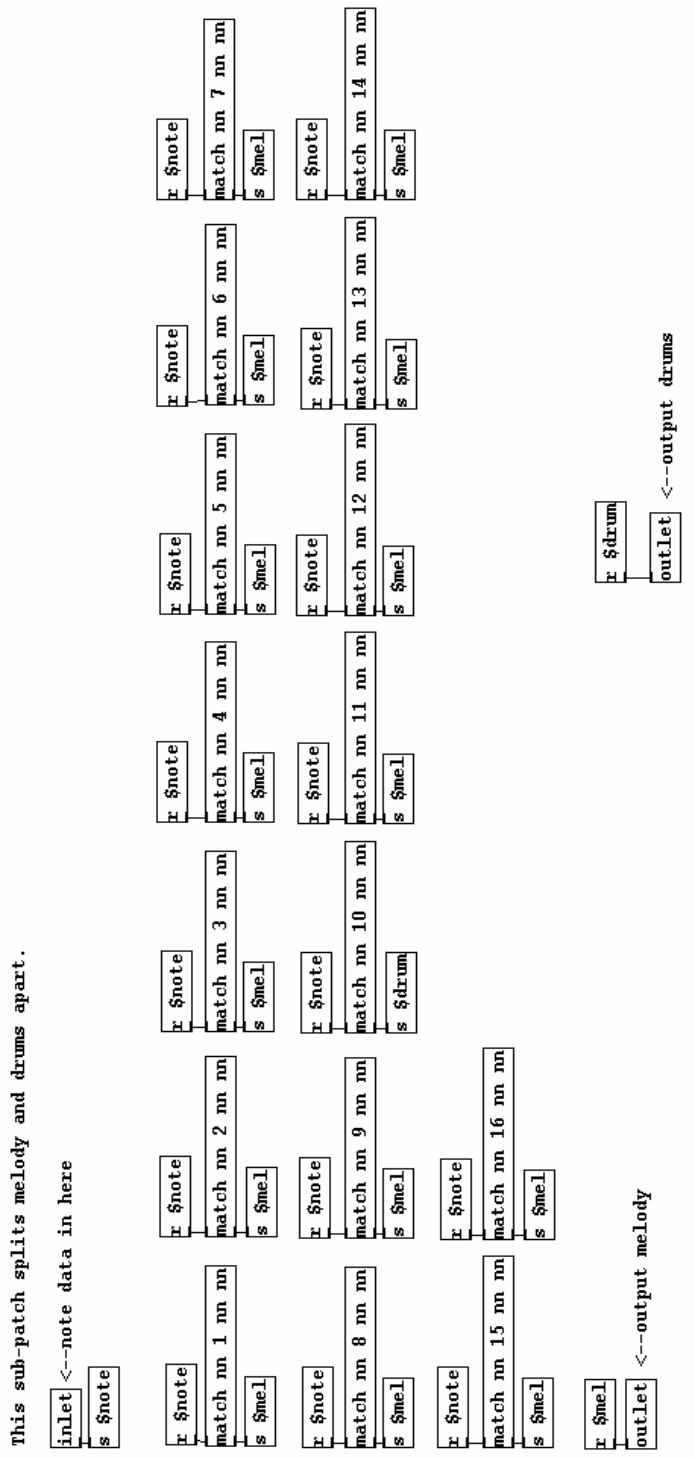


e) Activity Valence GEM window (Friberg, 2004)

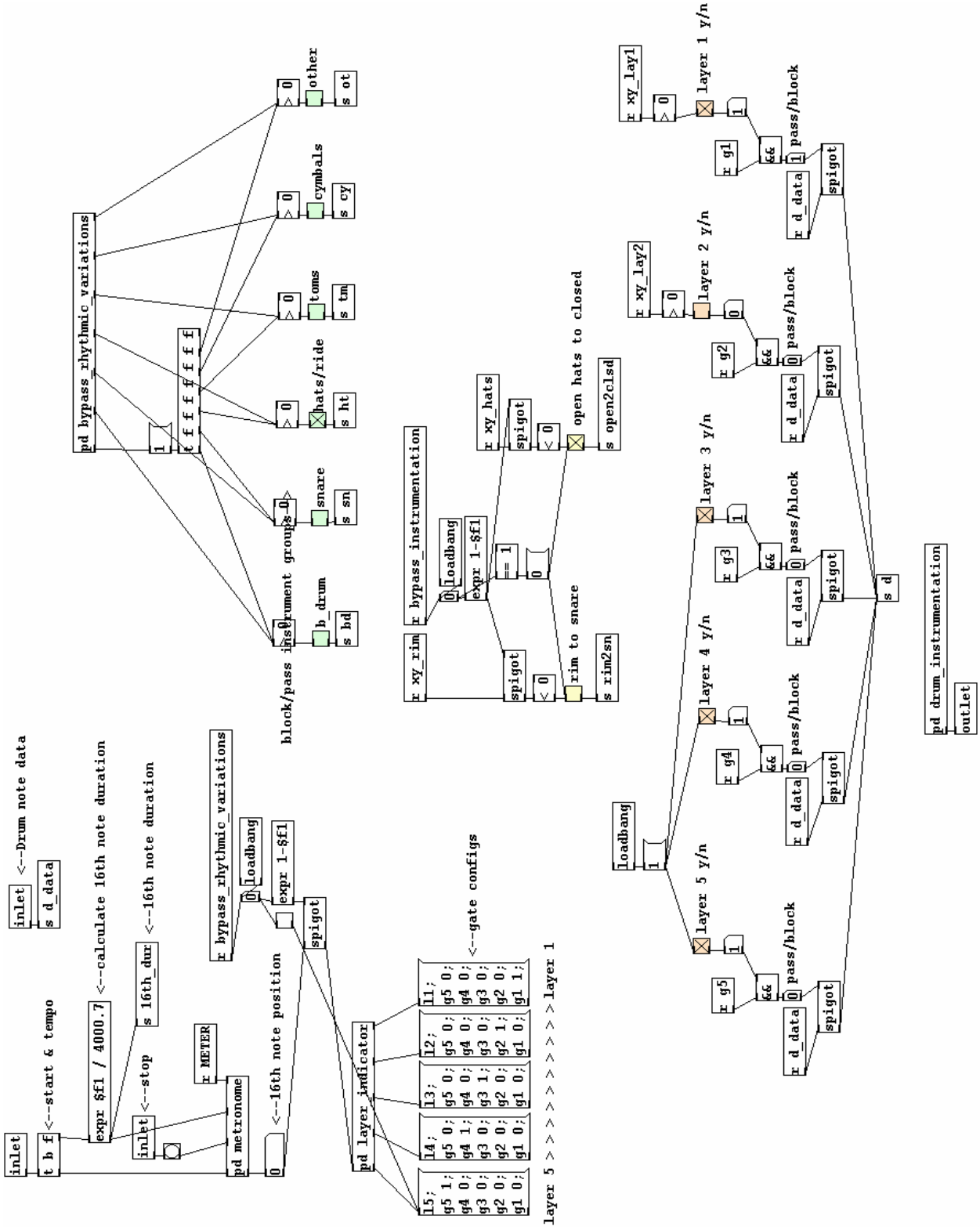
f) sequencer


note data in here
r NOTE
pd channel_filter
<--splits melody(L) and drums(R) apart
r TEMPO
r ustop
r stop
unpack f f
pd drum_filter
unpack f f f f
moses 11
pack f f f f
r CHORD
r KEY
pd bass
pd acco
unpack f f f f
pack f f f f
pd mute_control
pd instrumentation
r mode_type
r KEY
pd mode_manipulator
loadbang
unpack f f f f
0
61
pipe f f f 1000
100
noteout
note on
note off
read header info
r FILETYPE
print FILETYPE
r VERSION
print VERSION
r HEADERSTART
print HEADERSTART
r SCORENAME
print SCORENAME
r KEY
print KEY
r TRACKNAME
print TRACKNAME
r SYNTH
print SYNTH
r TRACKDELAY
print TRACKDELAY
r HEADEREND
r BAR
outlet
r METER
s METER
r PROGRAM
print PROGRAM
unpack f f
10
1
pgmout
<--program change
r VOL
<-pdm params
r DT
r DSL
r DART
loadbang
1
s mode_type_startup
inlet
s ustop
tempo 1
rewind
inlet
openpanel
read $1
inlet
qlist
r LOOPDELAY
s stop
print finished
delay 1000
inlet
switch

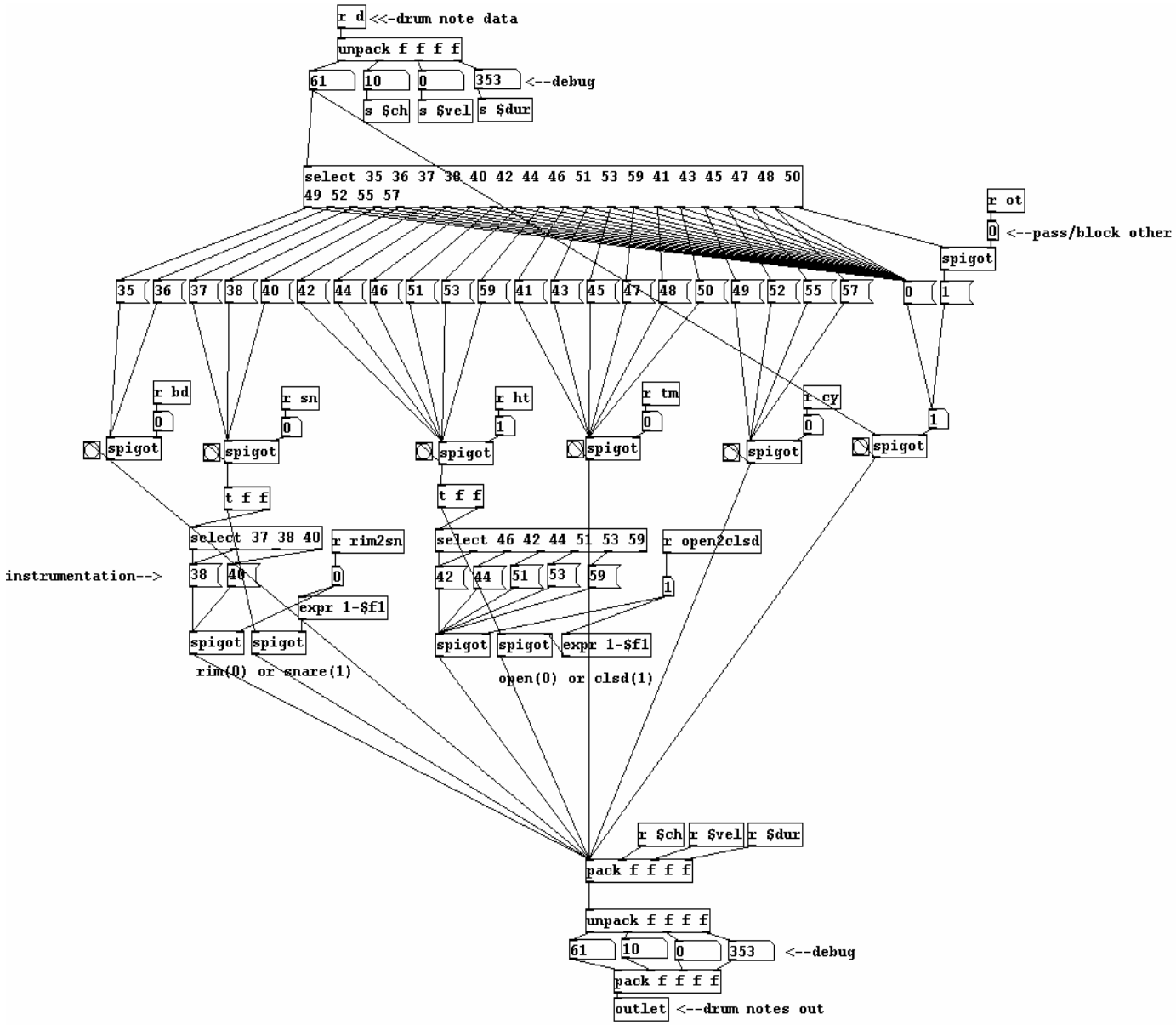
g) `channel_filter`



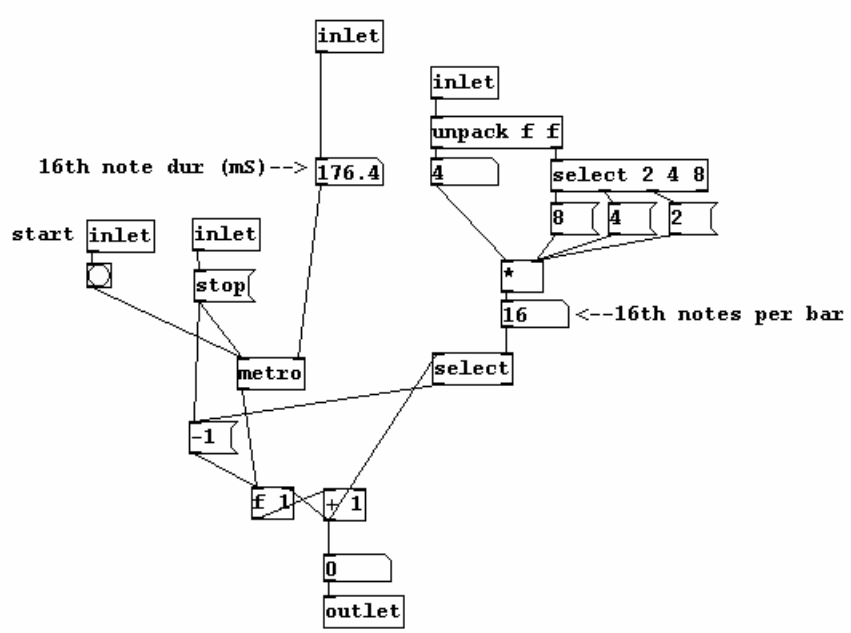
h) drum_filter



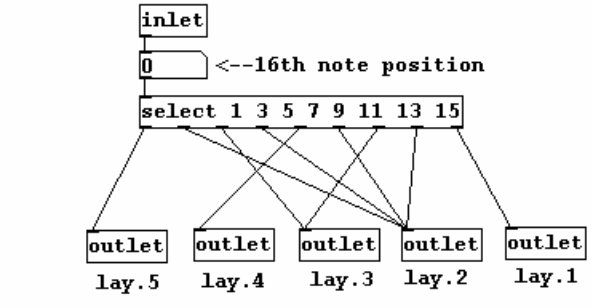
i) drum_instrumentation



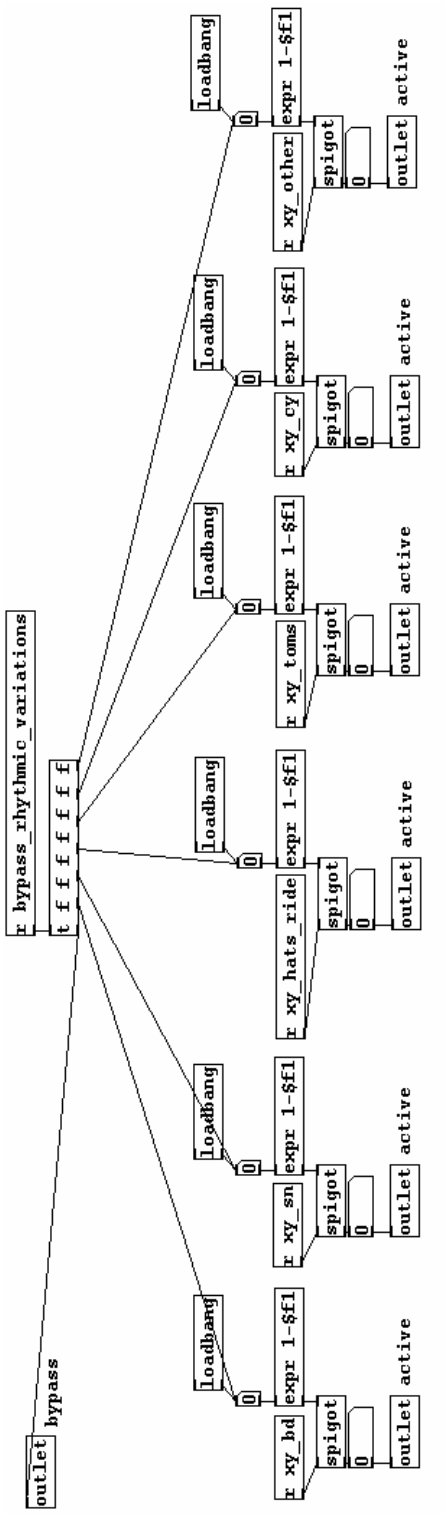
j) metronome



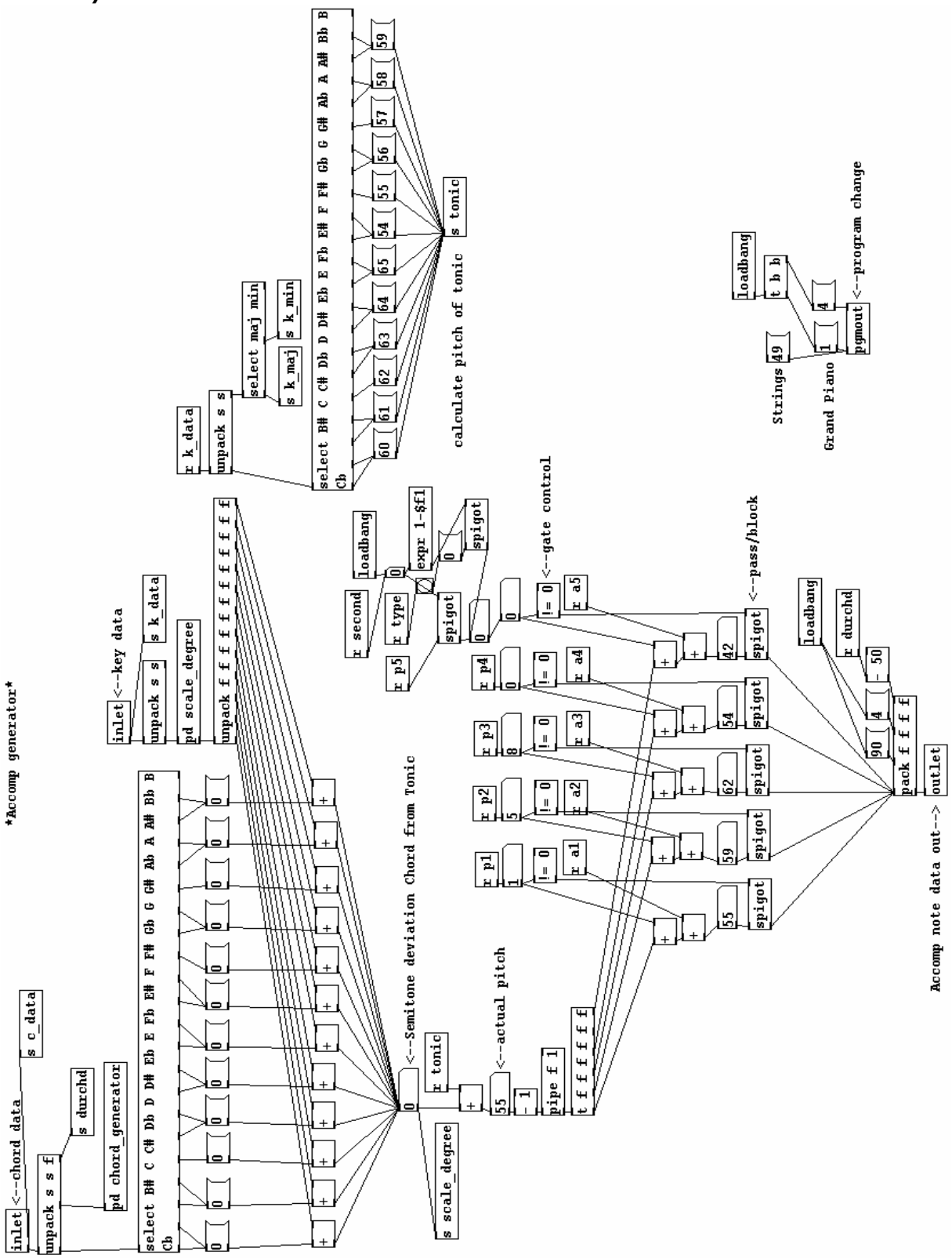
k) layer_indicator



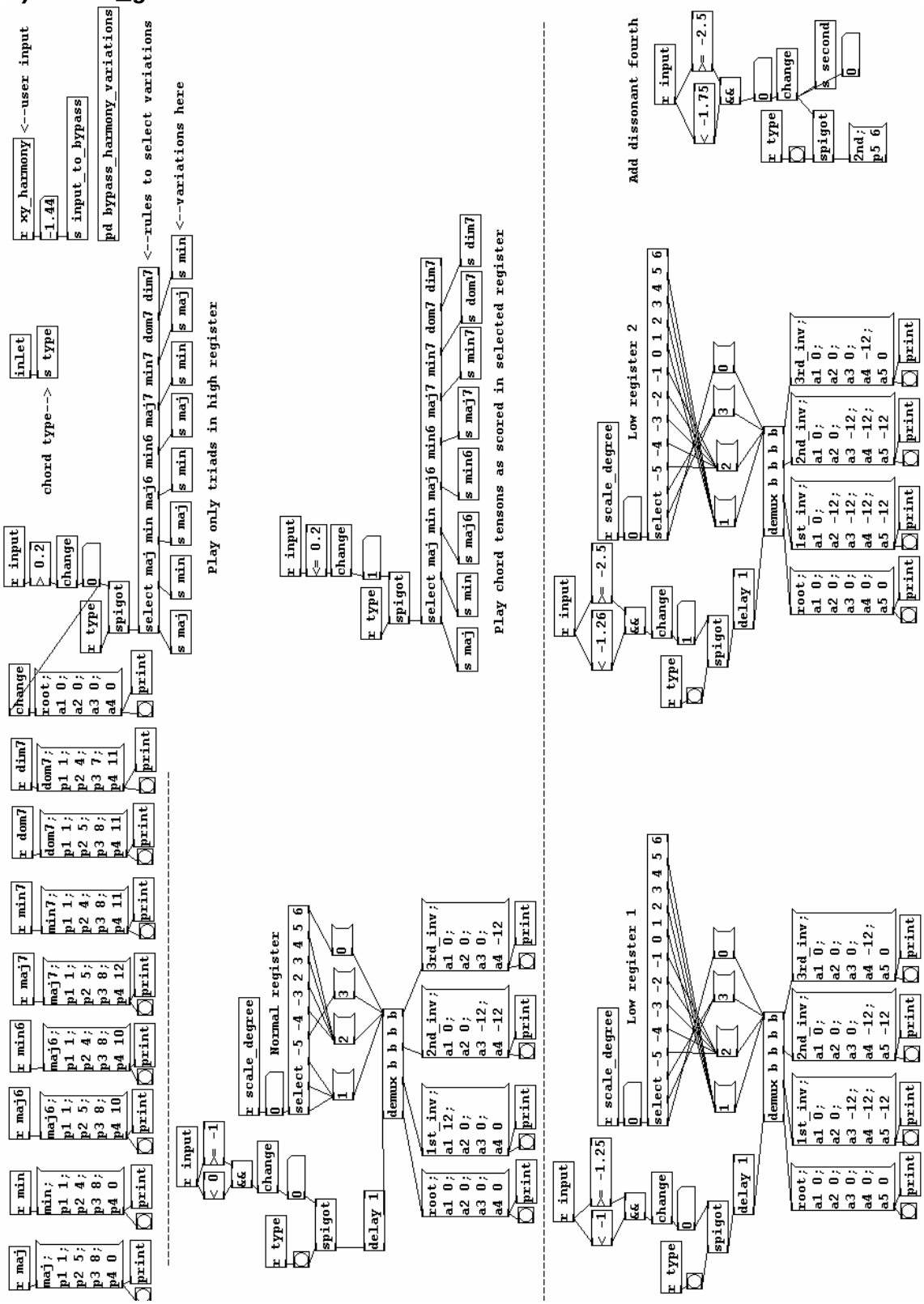
l) bypass_drum_instrumentation



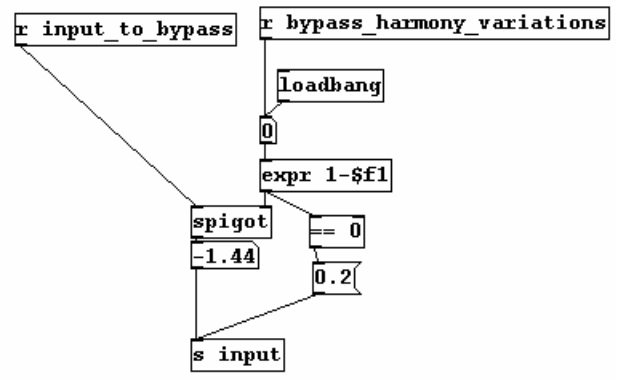
m) acco


Accomp generator
inlet <--chord data
s c_data
unpack s s f
s durchd
pd chord_generator
select B# C C# Db D D# Eb E Fb E# F F# Gb G G# Ab A A# Bb B
Cb
inlet <--key data
s k_data
unpack s s
pd scale_degree
unpack f f f f f f f f f f f f
r k_data
unpack s s
select maj min
s k_maj
s k_min
select B# C C# Db D D# Eb E Fb E# F F# Gb G G# Ab A A# Bb B
Cb
60 61 62 63 64 65 54 55 56 57 58 59
calculate pitch of tonic
s tonic
r second
Loadbang
r type
r p5
expr 1-$f1
spigot
<--gate control
r p4
r p3
r p2
r p1
!= 0
r a5
r a4
r a3
r a2
r a1
<--pass/block
Strings 49
Grand Piano 1
Loadbang
t b b
4
pgmout
<--program change
Loadbang
r durchd
- 50
90
4
pack f f f f f
outlet
Accomp note data out-->
<--Semitone deviation Chord from Tonic
r tonic
s scale_degree
<--actual pitch
55
- 1
pipe f 1
t f f f f f f

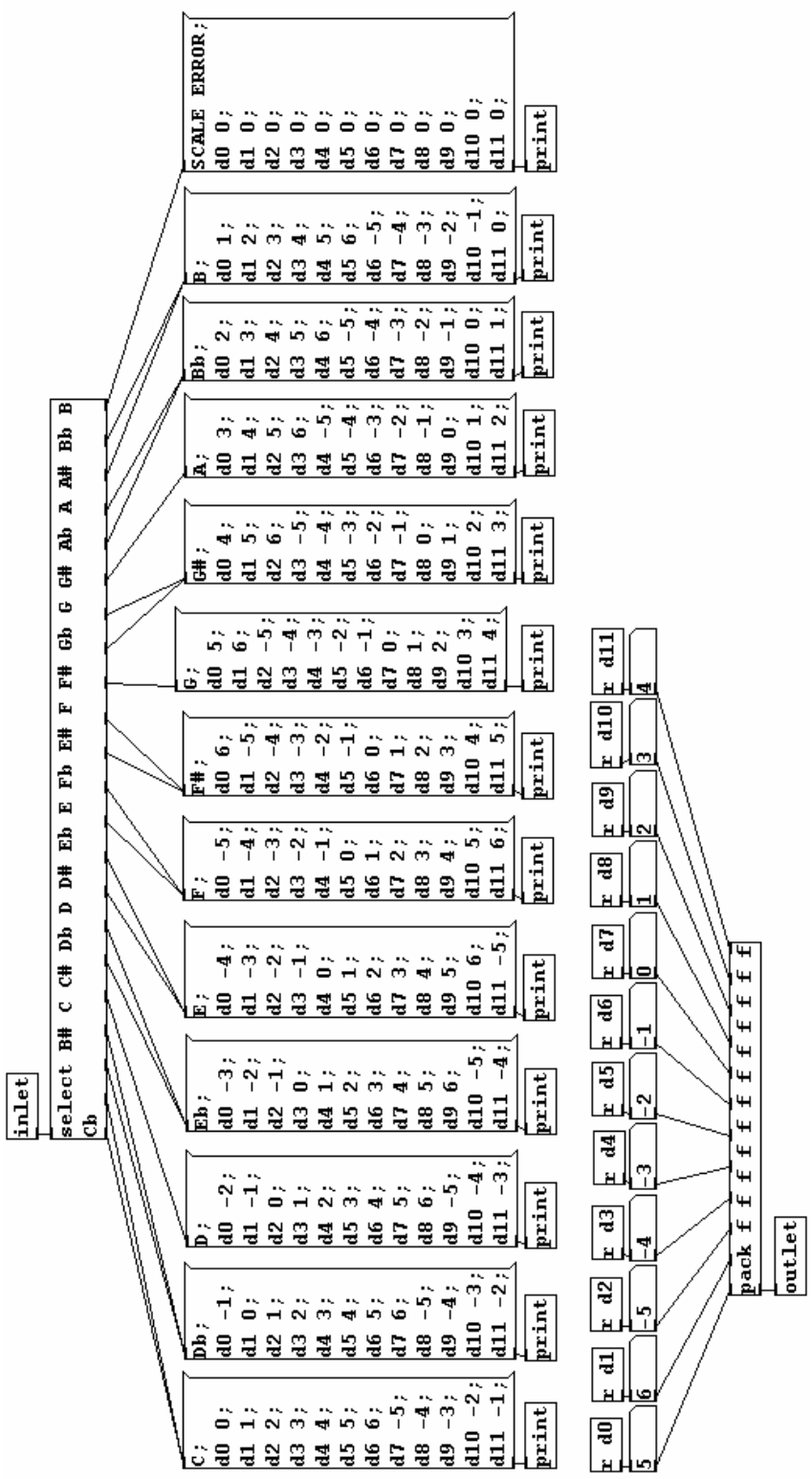
n) chord_generator



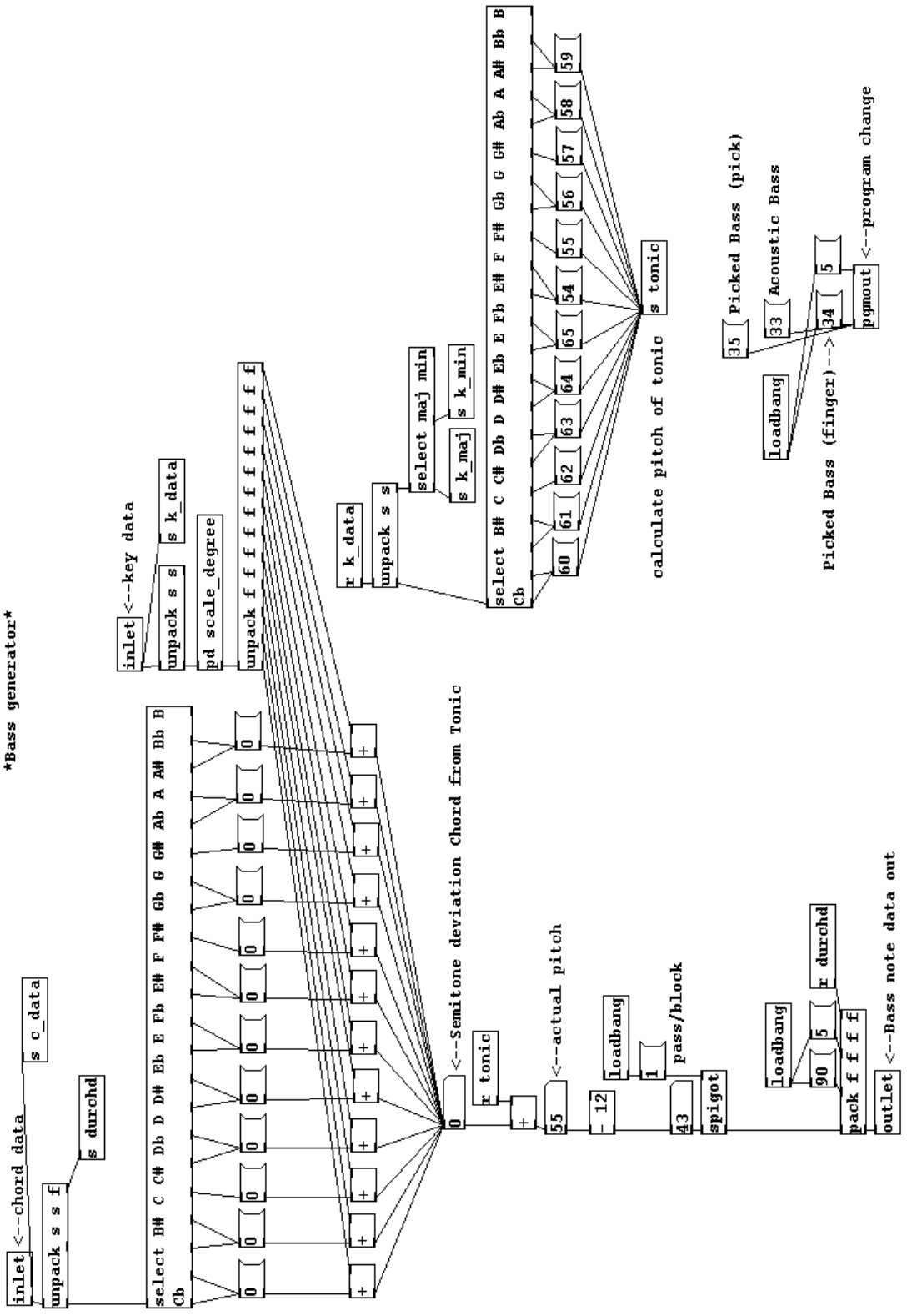
o) bypass_harmony_variations



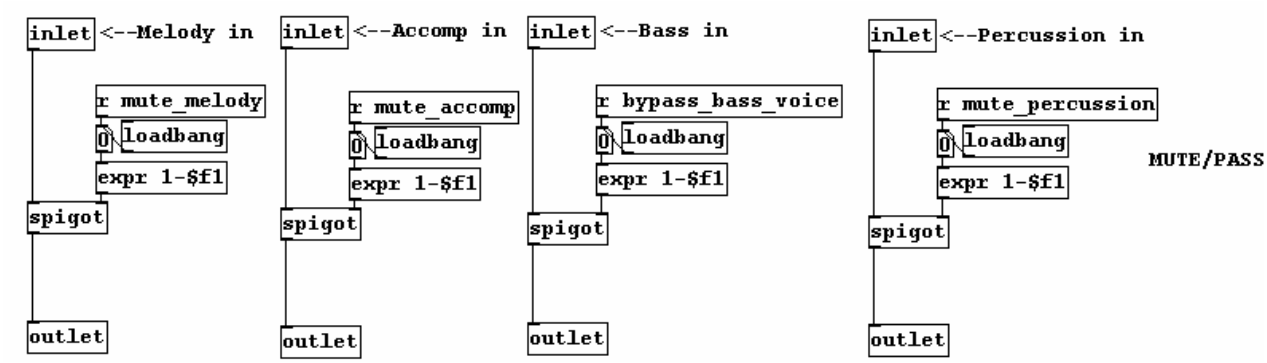
p) scale_degree



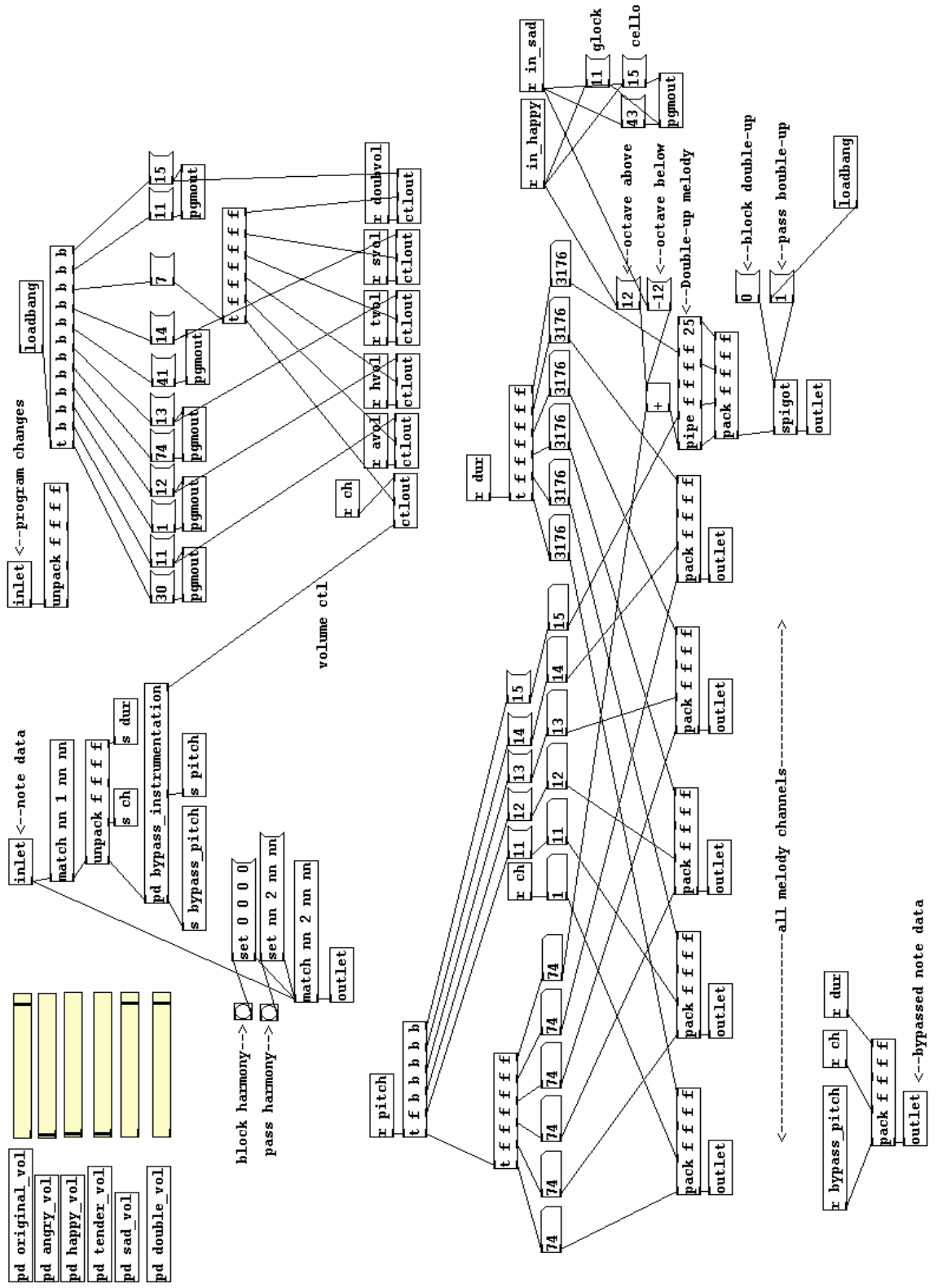
q) bass


Bass generator
inlet <--chord data
unpack s s f
s c_data
s durchd
select B# C C# Db D D# Eb E Fb E# F F# Gb G G# Ab A A# Bb B Cb
inlet <--key data
unpack s s
s k_data
pd scale_degree
unpack f f f f f f f f f f f f
r k_data
unpack s s
select maj min
s k_maj
s k_min
select B# C C# Db D D# Eb E Fb E# F F# Gb G G# Ab A A# Bb B Cb
60 61 62 63 64 65 54 55 56 57 58 59
calculate pitch of tonic
s tonic
35 Picked Bass (pick)
33 Acoustic Bass
loadbang
Picked Bass (finger)-->
34
5
pgmout <--program change
0 <--Semitone deviation Chord from Tonic
r tonic
+
55 <--actual pitch
- 12
loadbang
1
pass/block
43
spigot
loadbang
90
5
r durchd
pack f f f f
outlet <--Bass note data out

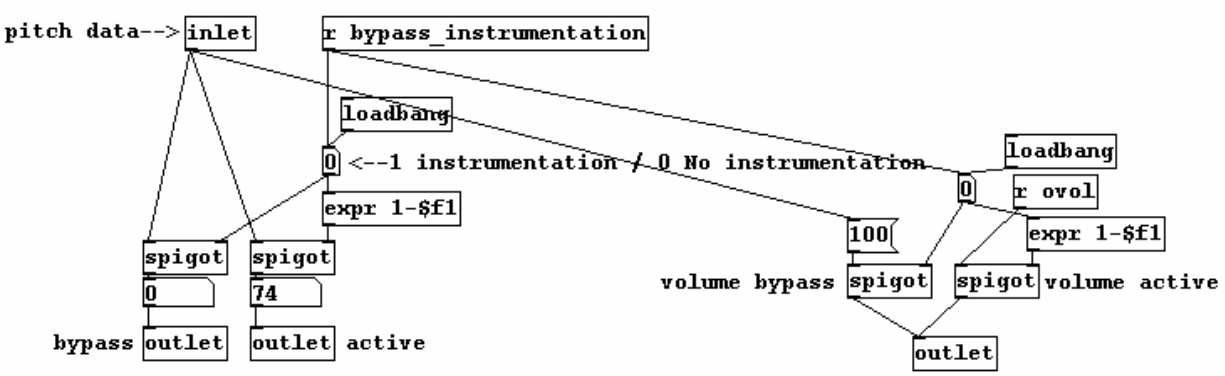
r) mute_control



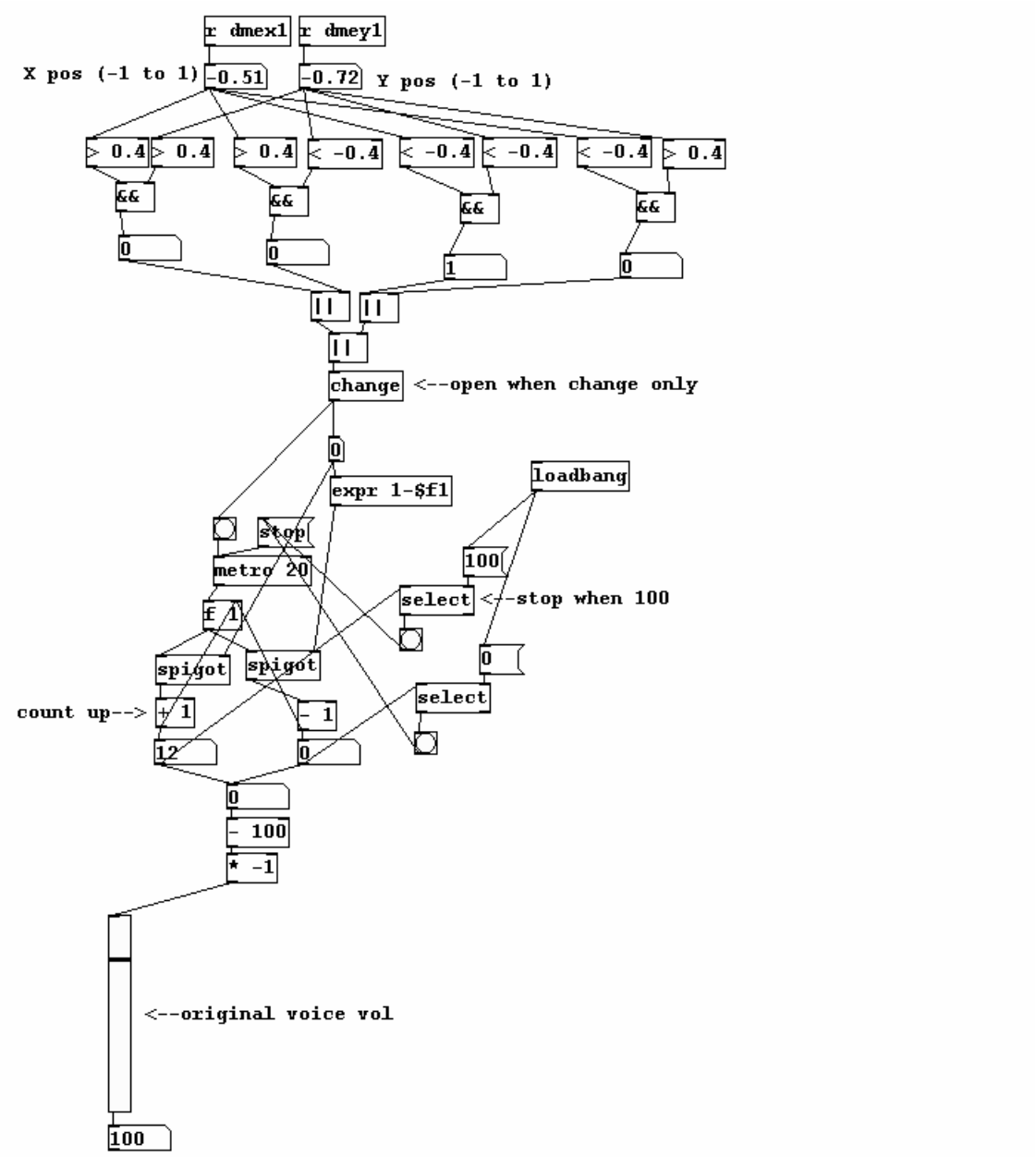
s) instrumentation



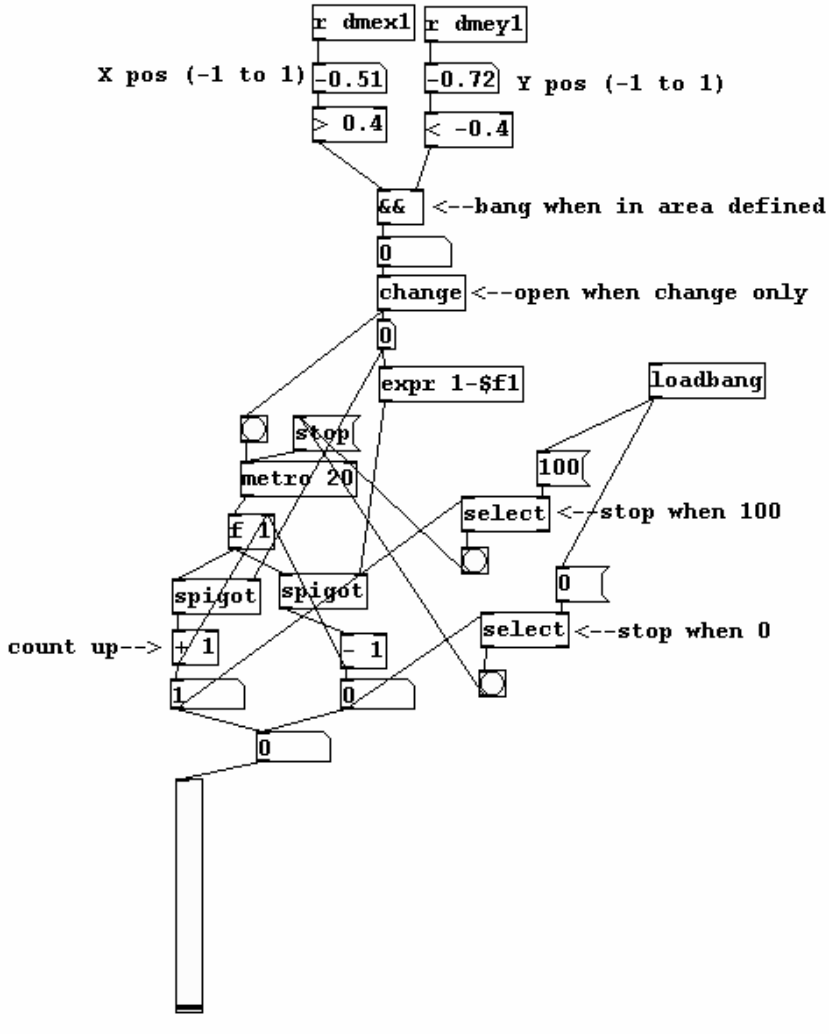
t) bypass_instrumentation



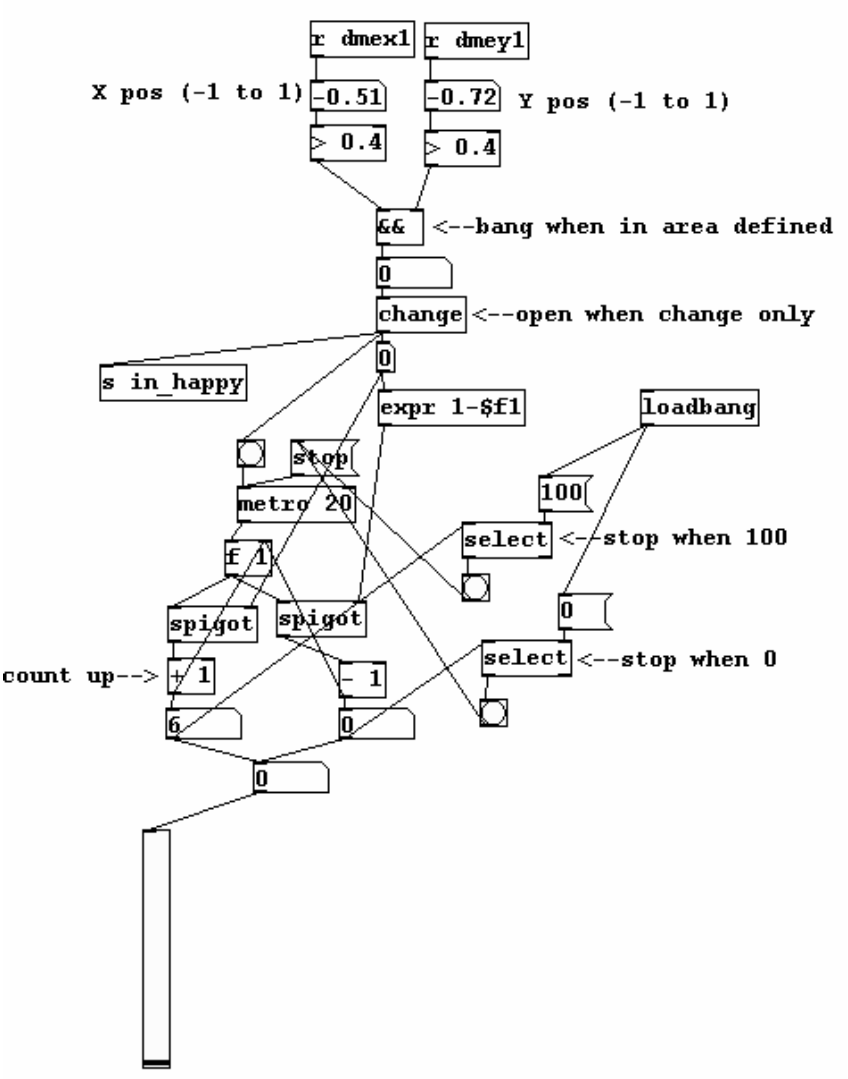
u) original_vol



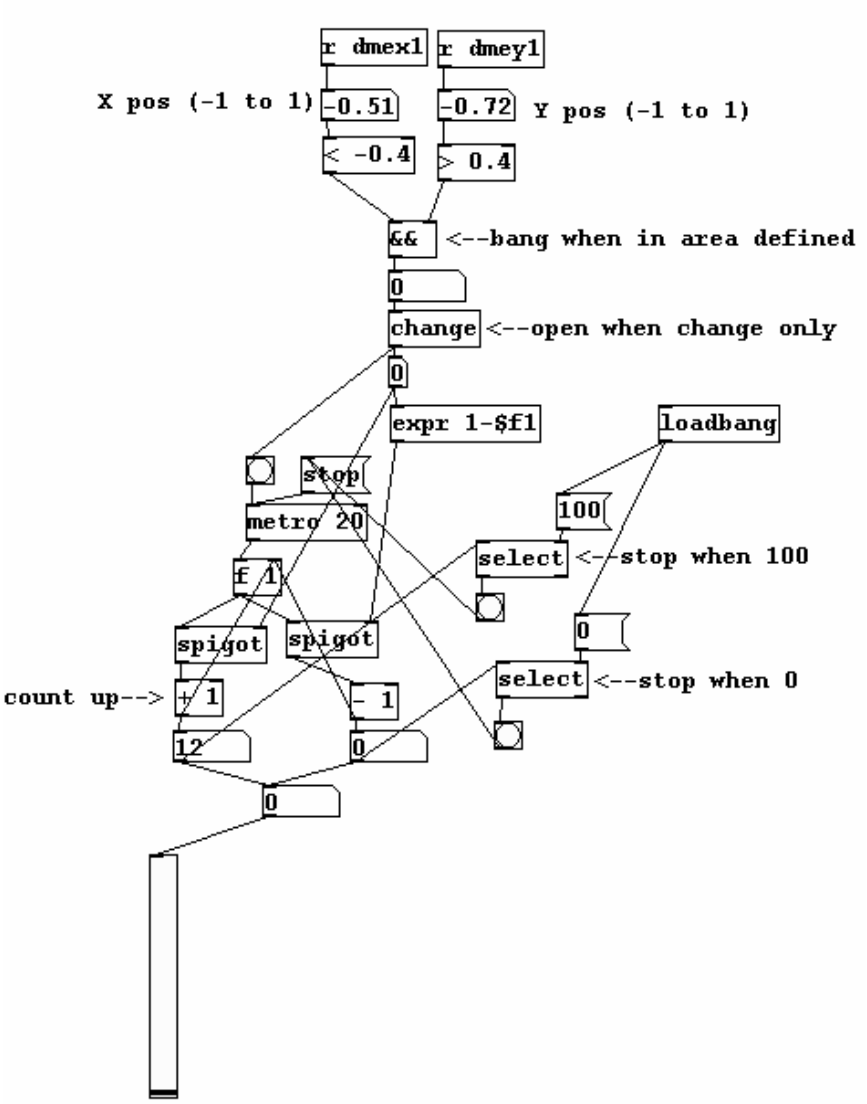
v) angry_vol


r dmex1
r dmey1
X pos (-1 to 1)
-0.51
-0.72
Y pos (-1 to 1)
> 0.4
< -0.4
&&
<--bang when in area defined
0
change
<--open when change only
0
expr 1-$f1
loadbang
stop
100
metro 20
select
<--stop when 100
f 1
0
spigot
spigot
select
<--stop when 0
count up-->
+ 1
- 1
1
0
0

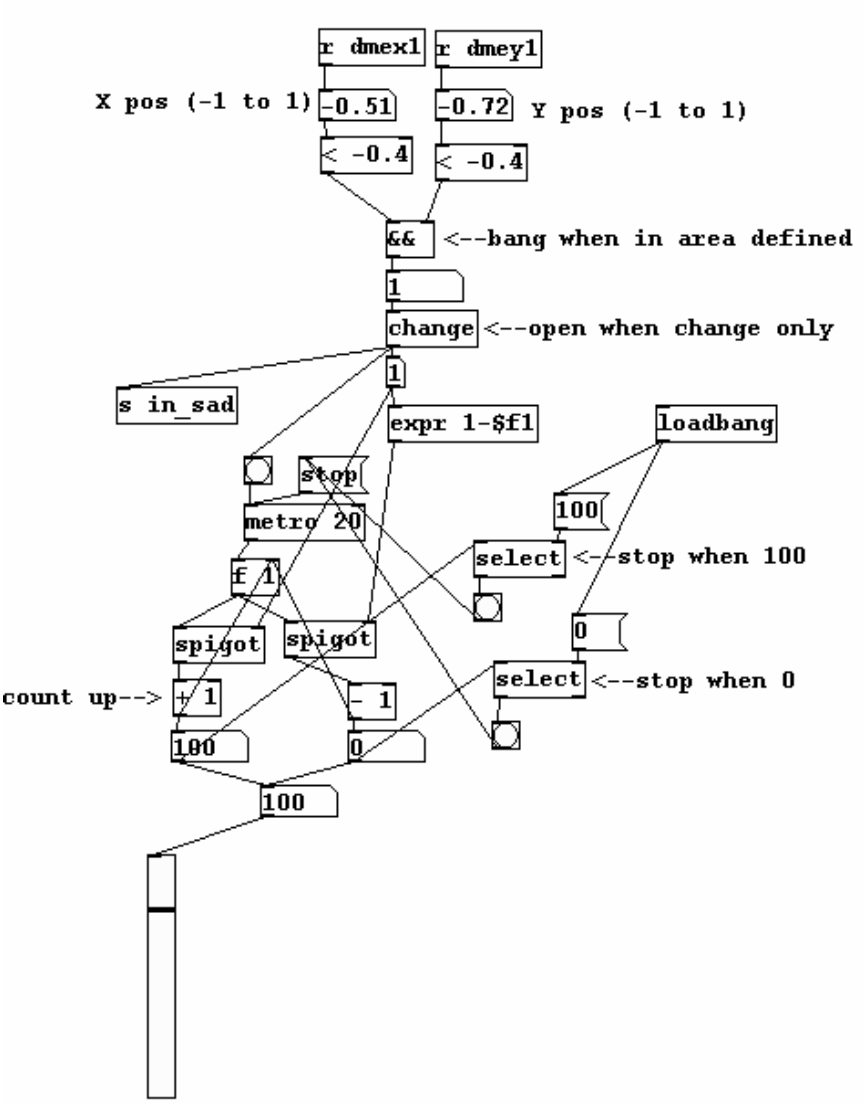
w) happy_vol


r dmex1
r dmey1
X pos (-1 to 1)
-0.51
-0.72
Y pos (-1 to 1)
> 0.4
> 0.4
&&
<--bang when in area defined
0
change
<--open when change only
0
s in_happy
expr 1-$f1
loadbang
stop
100
metro 20
select
<--stop when 100
f 1
0
spigot
spigot
select
<--stop when 0
count up-->
+ 1
- 1
6
0
0

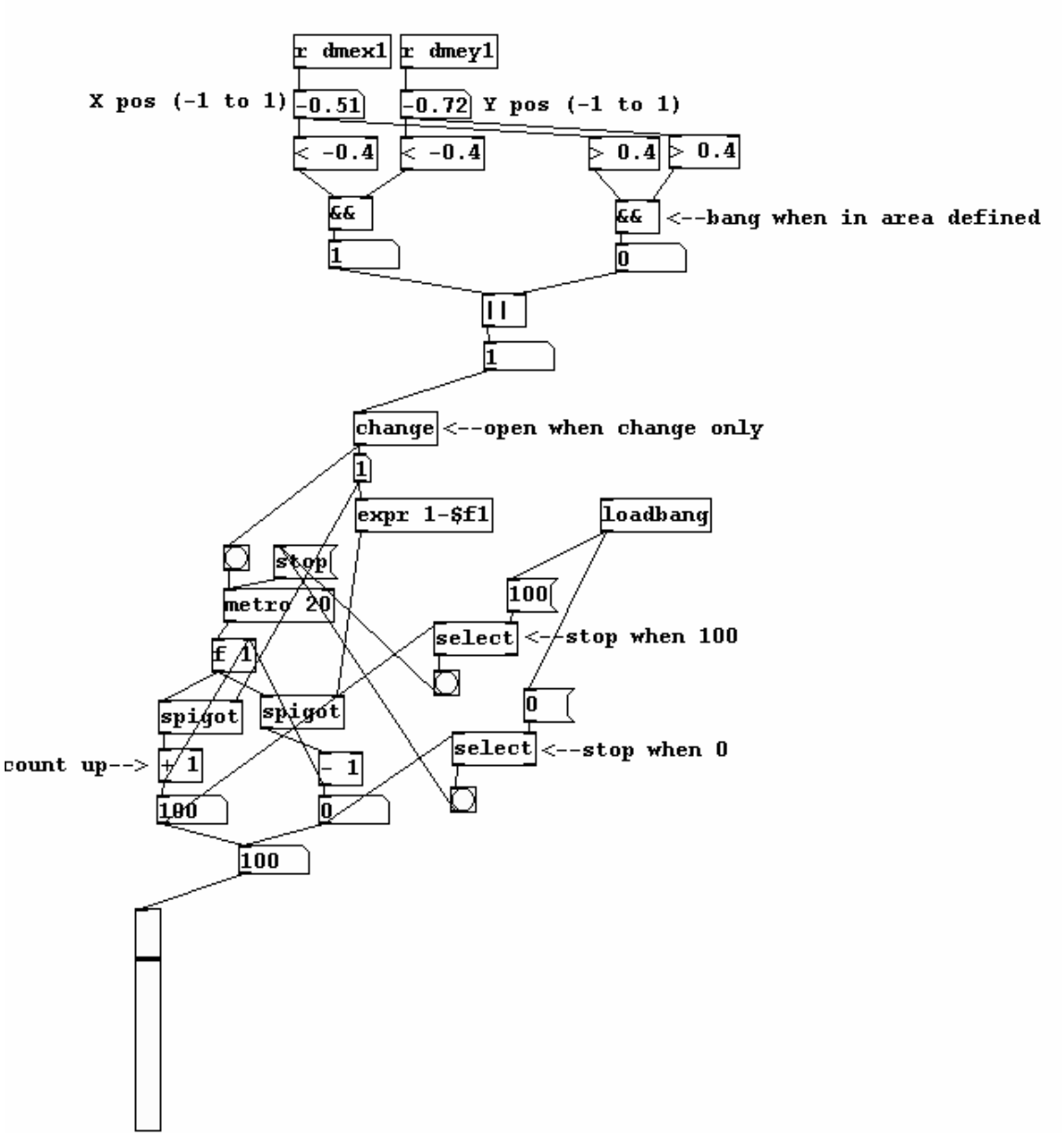
x) tender_vol


r dmex1
r dmey1
X pos (-1 to 1)
-0.51
-0.72
Y pos (-1 to 1)
< -0.4
> 0.4
&&
<--bang when in area defined
0
change
<--open when change only
0
expr 1-$f1
loadbang
stop
100
metro 20
select
<--stop when 100
f 1
0
spigot
spigot
select
<--stop when 0
count up-->
+ 1
- 1
12
0
0

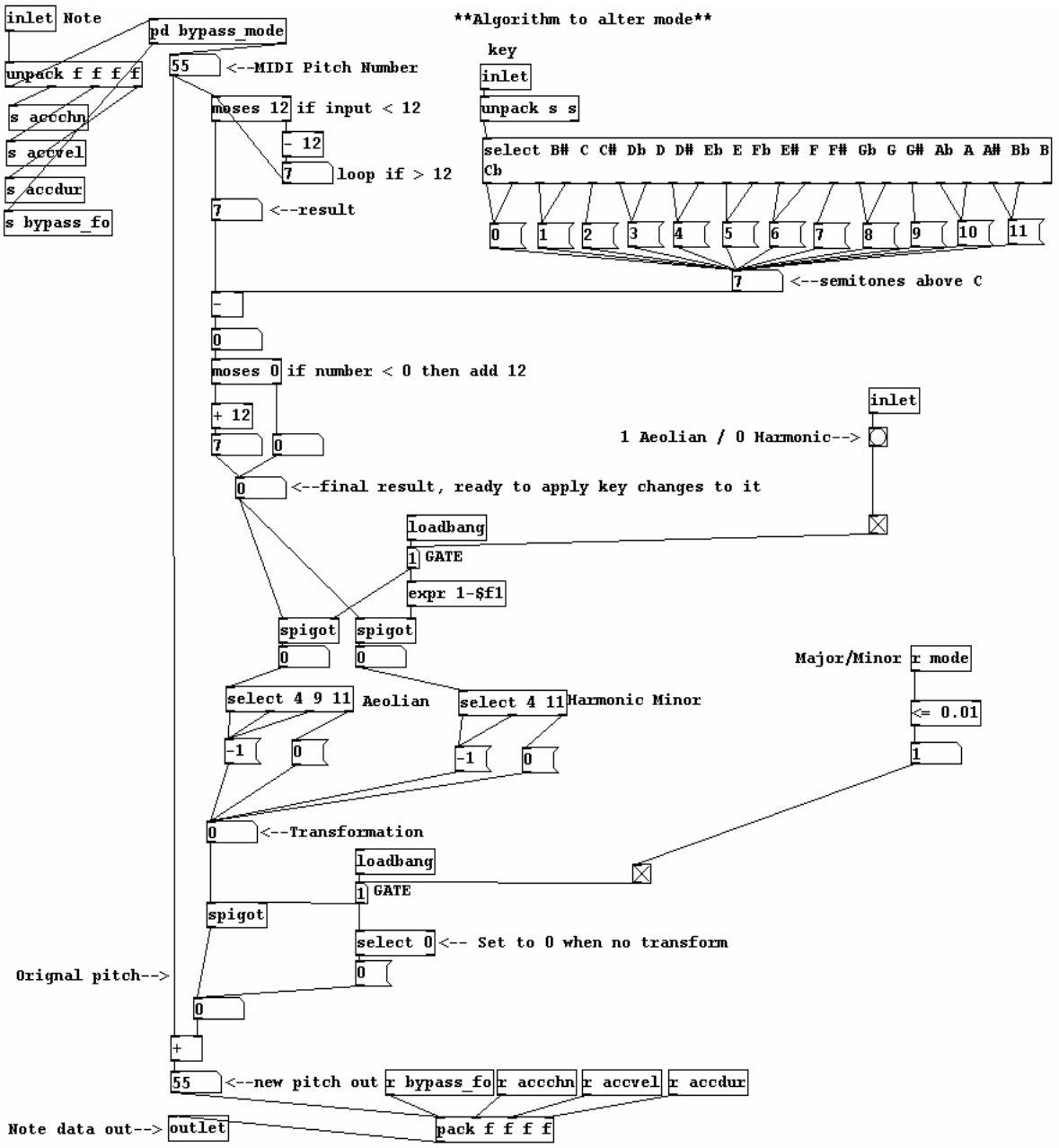
y) sad_vol


r dmex1
r dmey1
X pos (-1 to 1)
-0.51
-0.72
Y pos (-1 to 1)
< -0.4
< -0.4
&&
<--bang when in area defined
1
change
<--open when change only
1
s in_sad
expr 1-$f1
loadbang
stop
100
metro 20
select
<--stop when 100
f 1
0
spigot
spigot
select
<--stop when 0
count up-->
+ 1
- 1
100
0
100

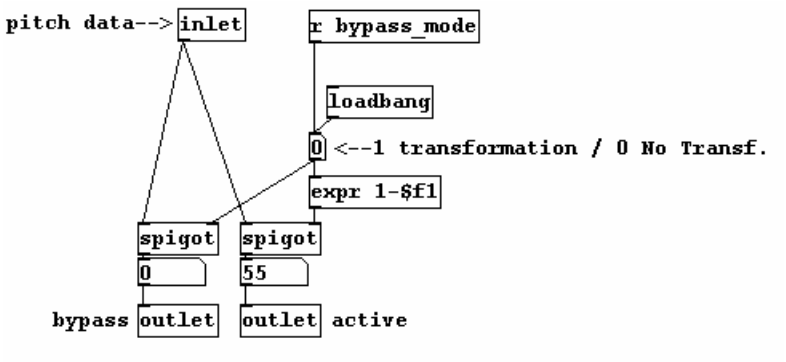
z) double_vol


r dmex1
r dmey1
X pos (-1 to 1)
-0.51
-0.72
Y pos (-1 to 1)
< -0.4
< -0.4
> 0.4
> 0.4
&&
&&
<--bang when in area defined
1
0
||
1
change
<--open when change only
1
expr 1-$f1
loadbang
stop
metro 20
100
select
<--stop when 100
f 1
0
spigot
spigot
select
<--stop when 0
count up-->
+ 1
- 1
100
0
100

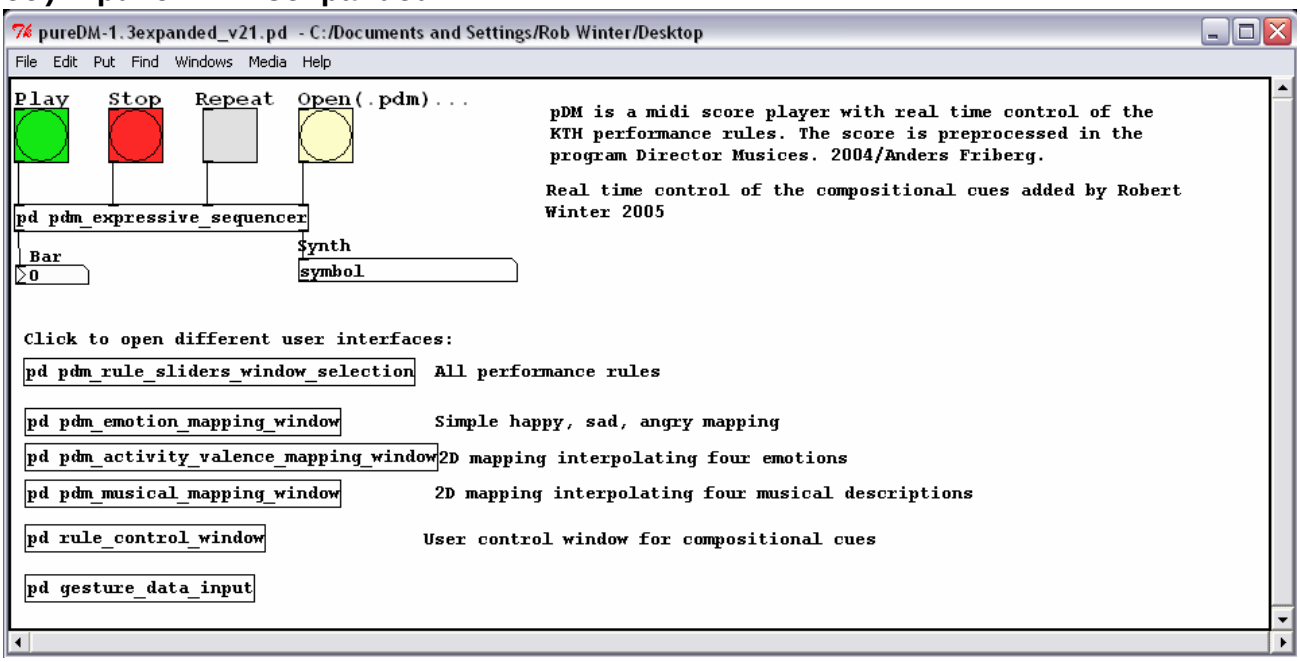
aa) mode_manipulator



bb) bypass_mode

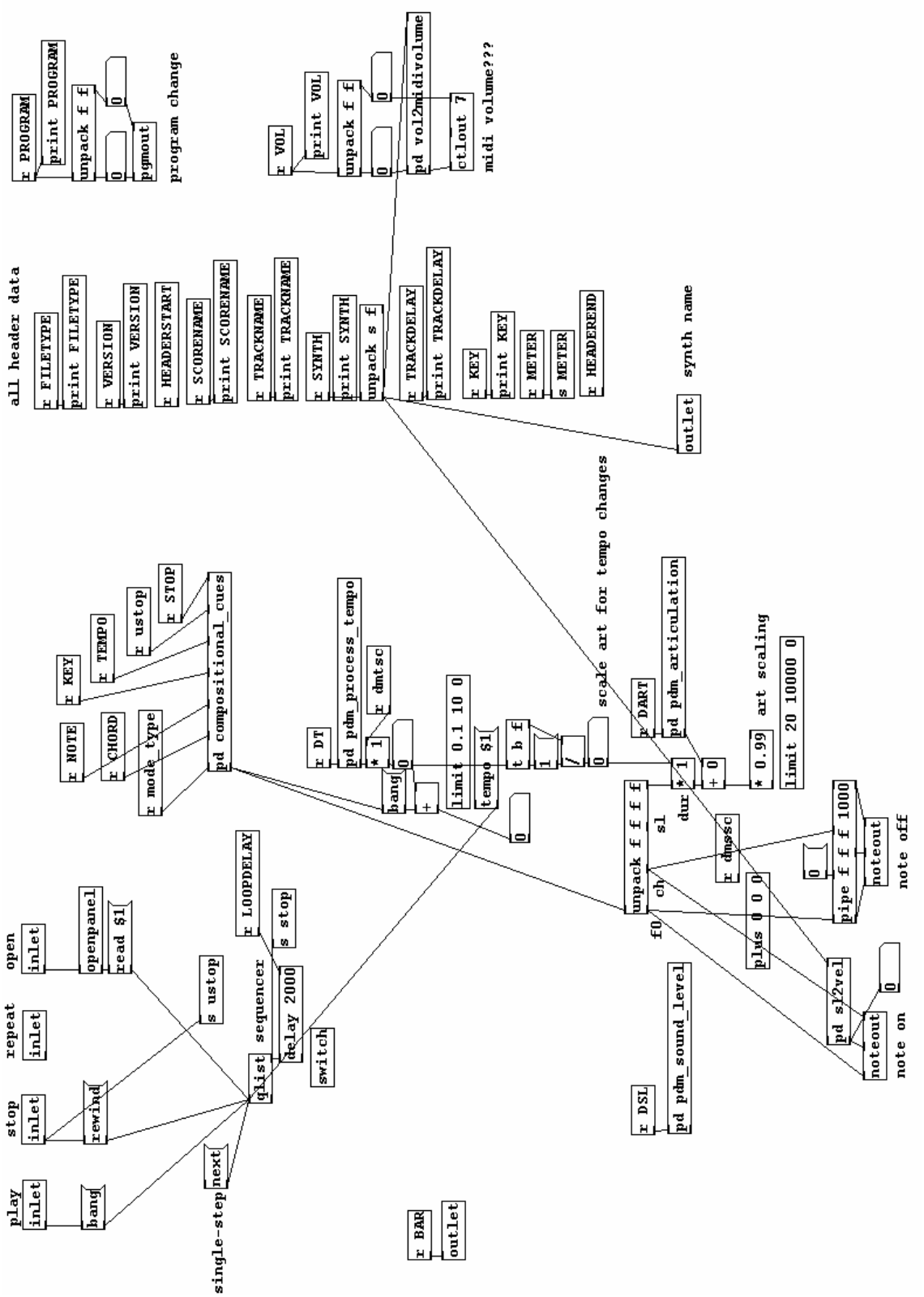


cc) pureDM-1.3expanded



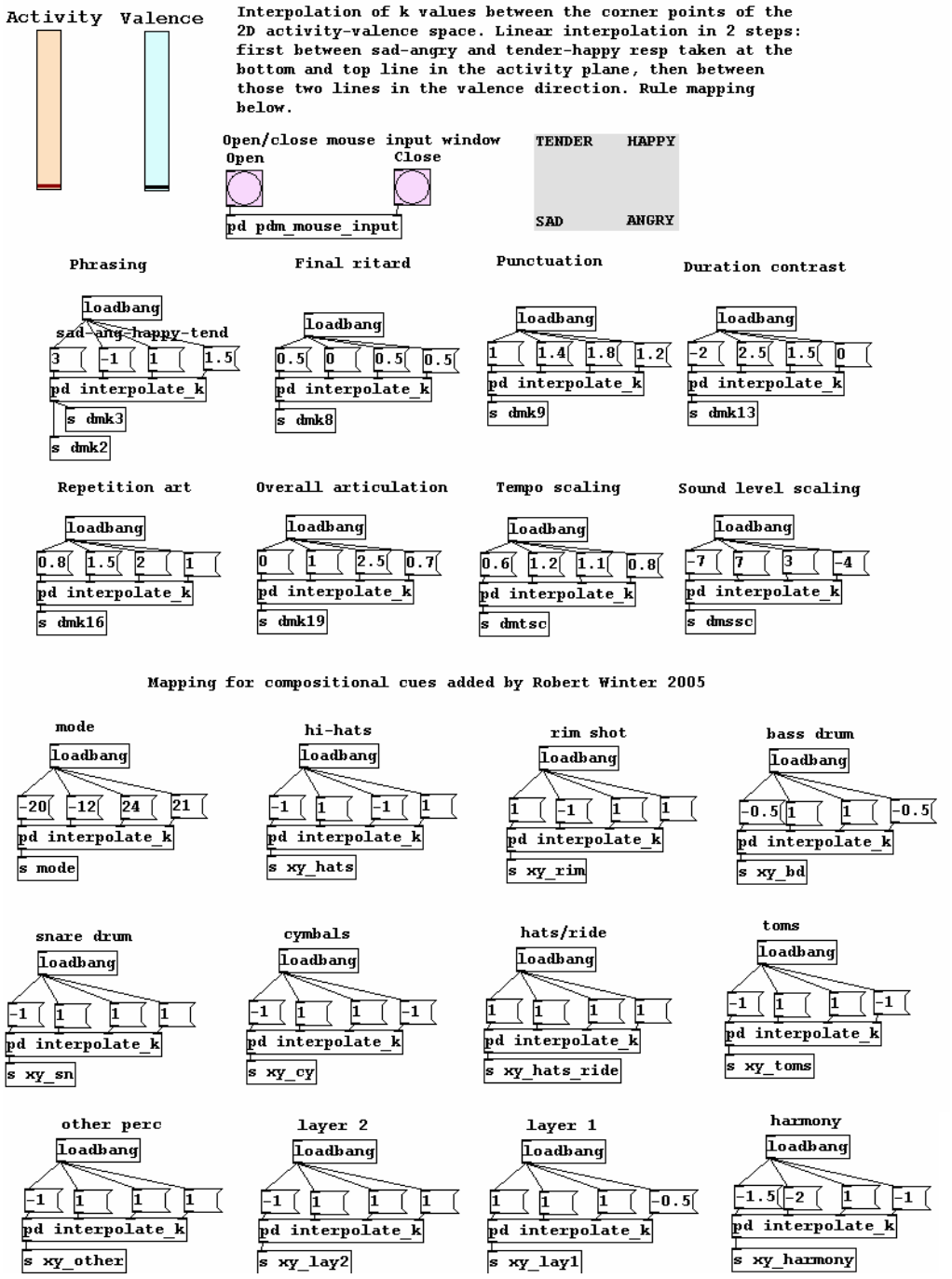
(Friberg, 2004 & Winter, 2005)

dd) expressive_sequencer



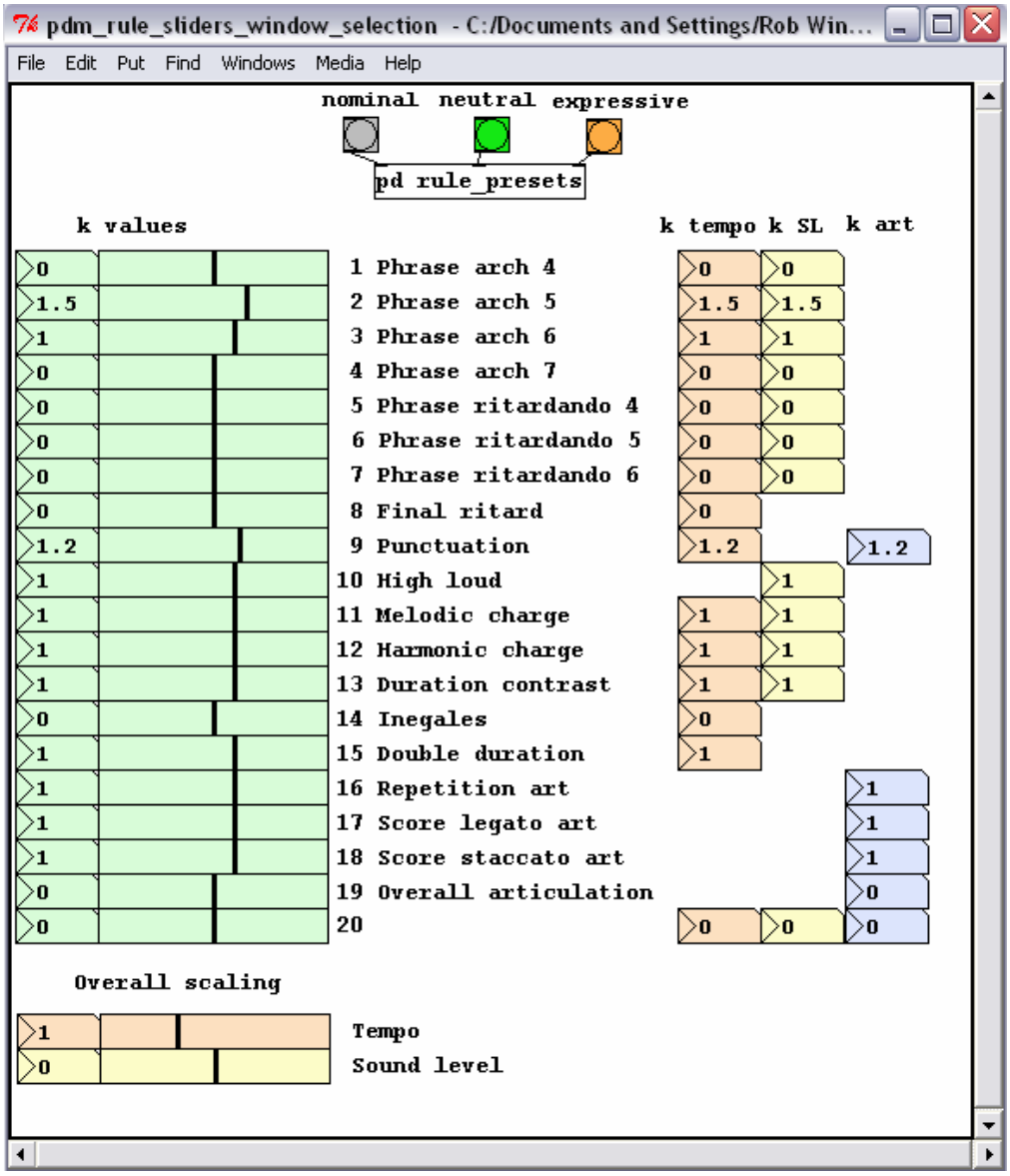
(Friberg, 2004 & Winter, 2005)

ee) pdm_activity_valence_mapping_window



(Friberg, 2004 & Winter, 2005)

ff) pdm_rule_sliders_window_selection



(Friberg, 2004)

D. General MIDI Level 1 Instrument List and Drum Map

GM1 Instrument List

PC#	Instrument	PC#	Instrument
1.	Acoustic Grand Piano	65.	Soprano Sax
2.	Bright Acoustic Piano	66.	Alto Sax
3.	Electric Grand Piano	67.	Tenor Sax
4.	Honky-tonk Piano	68.	Baritone Sax
5.	Electric Piano 1	69.	Oboe
6.	Electric Piano 2	70.	English Horn
7.	Harpsichord	71.	Bassoon
8.	Clavi	72.	Clarinet
9.	Celesta	73.	Piccolo
10.	Glockenspiel	74.	Flute
11.	Music Box	75.	Recorder
12.	Vibraphone	76.	Pan Flute
13.	Marimba	77.	Blown Bottle
14.	Xylophone	78.	Shakuhachi
15.	Tubular Bells	79.	Whistle
16.	Dulcimer	80.	Ocarina
17.	Drawbar Organ	81.	Lead 1 (square)
18.	Percussive Organ	82.	Lead 2 (sawtooth)
19.	Rock Organ	83.	Lead 3 (calliope)
20.	Church Organ	84.	Lead 4 (chiff)
21.	Reed Organ	85.	Lead 5 (charang)
22.	Accordion	86.	Lead 6 (voice)
23.	Harmonica	87.	Lead 7 (fifths)
24.	Tango Accordion	88.	Lead 8 (bass + lead)
25.	Acoustic Guitar (nylon)	89.	Pad 1 (new age)
26.	Acoustic Guitar (steel)	90.	Pad 2 (warm)
27.	Electric Guitar (jazz)	91.	Pad 3 (polysynth)
28.	Electric Guitar (clean)	92.	Pad 4 (choir)
29.	Electric Guitar (muted)	93.	Pad 5 (bowed)
30.	Overdriven Guitar	94.	Pad 6 (metallic)
31.	Distortion Guitar	95.	Pad 7 (halo)
32.	Guitar harmonics	96.	Pad 8 (sweep)
33.	Acoustic Bass	97.	FX 1 (rain)
34.	Electric Bass (finger)	98.	FX 2 (soundtrack)
35.	Electric Bass (pick)	99.	FX 3 (crystal)
36.	Fretless Bass	100.	FX 4 (atmosphere)
37.	Slap Bass 1	101.	FX 5 (brightness)
38.	Slap Bass 2	102.	FX 6 (goblins)
39.	Synth Bass 1	103.	FX 7 (echoes)
40.	Synth Bass 2	104.	FX 8 (sci-fi)
41.	Violin	105.	Sitar
42.	Viola	106.	Banjo
43.	Cello	107.	Shamisen
44.	Contrabass	108.	Koto
45.	Tremolo Strings	109.	Kalimba
46.	Pizzicato Strings	110.	Bag pipe
47.	Orchestral Harp	111.	Fiddle
48.	Timpani	112.	Shanai
49.	String Ensemble 1	113.	Tinkle Bell
50.	String Ensemble 2	114.	Agogo
51.	SynthStrings 1	115.	Steel Drums

52. SynthStrings 2
53. Choir Aahs
54. Voice Oohs
55. Synth Voice
56. Orchestra Hit
57. Trumpet
58. Trombone
59. Tuba
60. Muted Trumpet
61. French Horn
62. Brass Section
63. SynthBrass 1
64. SynthBrass 2
116. Woodblock
117. Taiko Drum
118. Melodic Tom
119. Synth Drum
120. Reverse Cymbal
121. Guitar Fret Noise
122. Breath Noise
123. Seashore
124. Bird Tweet
125. Telephone Ring
126. Helicopter
127. Applause
128. Gunshot

GM1 Percussion Key Map

Key#	Drum Sound	Key#	Drum Sound
35	Acoustic Bass Drum	59	Ride Cymbal 2
36	Bass Drum 1	60	Hi Bongo
37	Side Stick	61	Low Bongo
38	Acoustic Snare	62	Mute Hi Conga
39	Hand Clap	63	Open Hi Conga
40	Electric Snare	64	Low Conga
41	Low Floor Tom	65	High Timbale
42	Closed Hi Hat	66	Low Timbale
43	High Floor Tom	67	High Agogo
44	Pedal Hi-Hat	68	Low Agogo
45	Low Tom	69	Cabasa
46	Open Hi-Hat	70	Maracas
47	Low-Mid Tom	71	Short Whistle
48	Hi Mid Tom	72	Long Whistle
49	Crash Cymbal 1	73	Short Guiro
50	High Tom	74	Long Guiro
51	Ride Cymbal 1	75	Claves
52	Chinese Cymbal	76	Hi Wood Block
53	Ride Bell	77	Low Wood Block
54	Tambourine	78	Mute Cuica
55	Splash Cymbal	79	Open Cuica
56	Cowbell	80	Mute Triangle
57	Crash Cymbal 2	81	Open Triangle
58	Vibraslap		

Source: Midi Manufactures Association Website:
http://www.midi.org/
Accessed 05/05/05

E. Creating and Compiling an External in Pure Data

Creating an External

For a useful guide, refer to Zmölnig: http://pd.iem.at/externals-HOWTO/

Compiling an External

Externals written in Windows use the C++ language. A single, or group of externals must be compiled into a single binary file called a library. The naming convention for Win32 is `my_lib.dll`. To create a library file for use within Pure Data the process I have used is as follows using MS Visual C++:

1. Download a template workspace to base your external upon. A good example is the `zexy.dsw` workspace available at: ftp://ftp.iem.at/pd/Externals/ZEXY/
2. Unzip the downloaded file and navigate zexy/src/zexy.dsw
3. Load Microsoft Visual C++ and open `zexy.dsw` from the above location.
4. Delete all zexy files from the lefthand window box.
5. Add your own `*.c` or `*.cpp` code that will describe your external to the project.

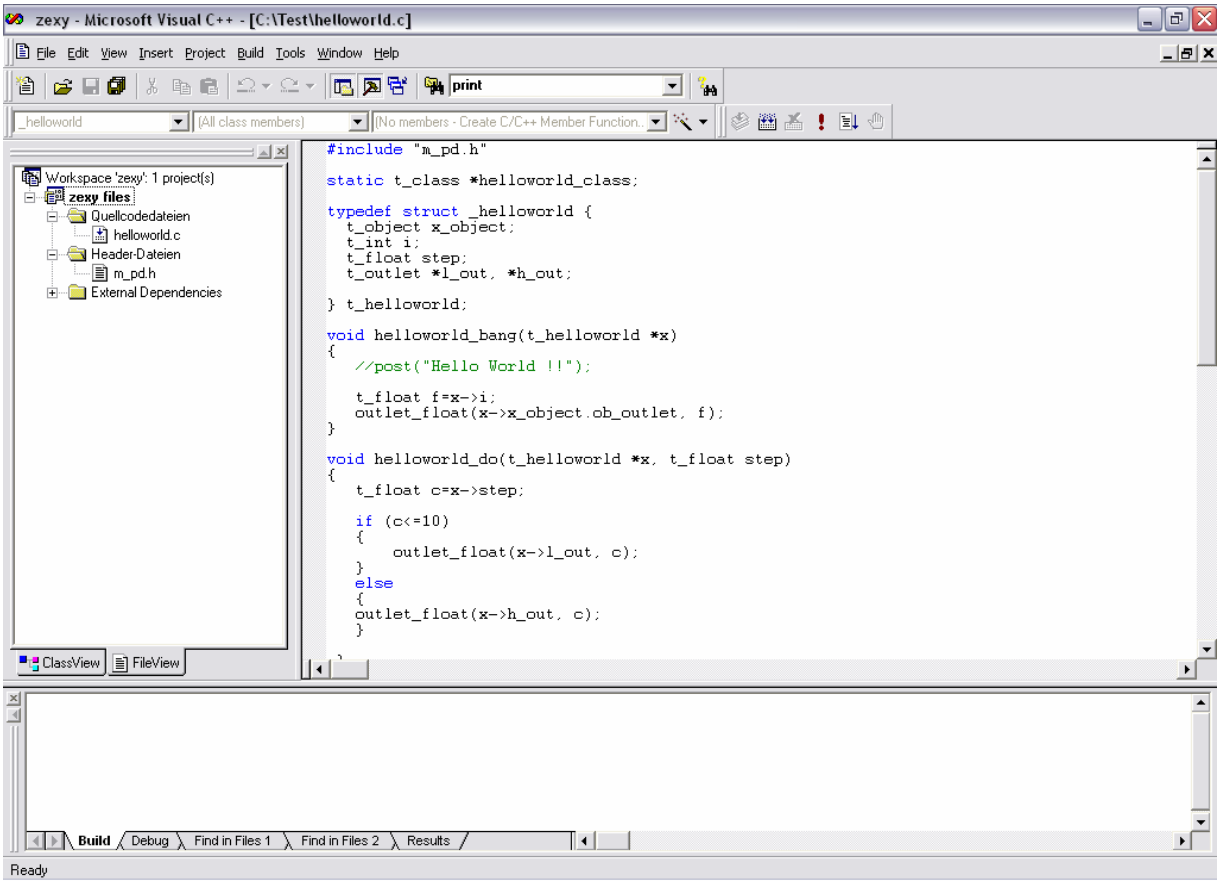


Figure E1: Main Microsoft Visual C++ Window

6. Save the `*.dsw` project to a new folder location (e.g. c:\externals) along with your original *.c code and the file `m_pd.h` found in the pure data folder. This is needed as all external class include this file as a header.
7. In Visual C++ navigate: Project\Settings
8. Click the C\C++ tab and place the cursor in the 'Preprocessor Definitions' field.
9. Rename '`ZEXY`' to '`MY_EXTERNAL`' where my_external is the name of your new external.

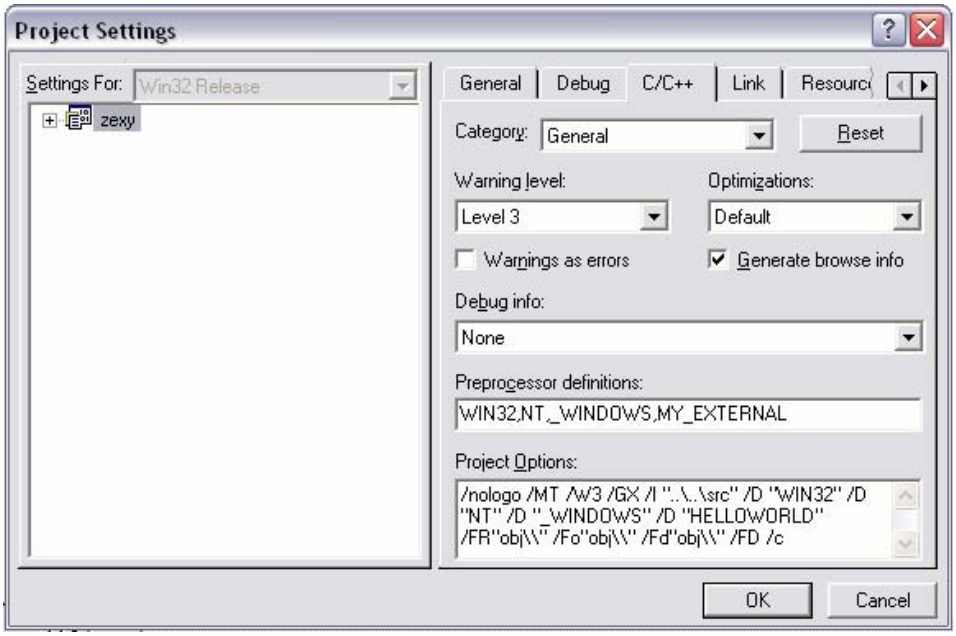


Figure E2: Project Settings Window

10. In the Category drop down menu select 'Precompiled Headers' and ensure that 'Not using precompiled headers' is selected.
11. Click the 'Link' tab and place the cursor in the 'Output file name' text field. Again, rename '`zexy.dll`' to '`my_external.dll`'.
12. Ensure that the path `pd.lib` is present at the end of the 'Object/Library modules:' text field. Add if necessary.

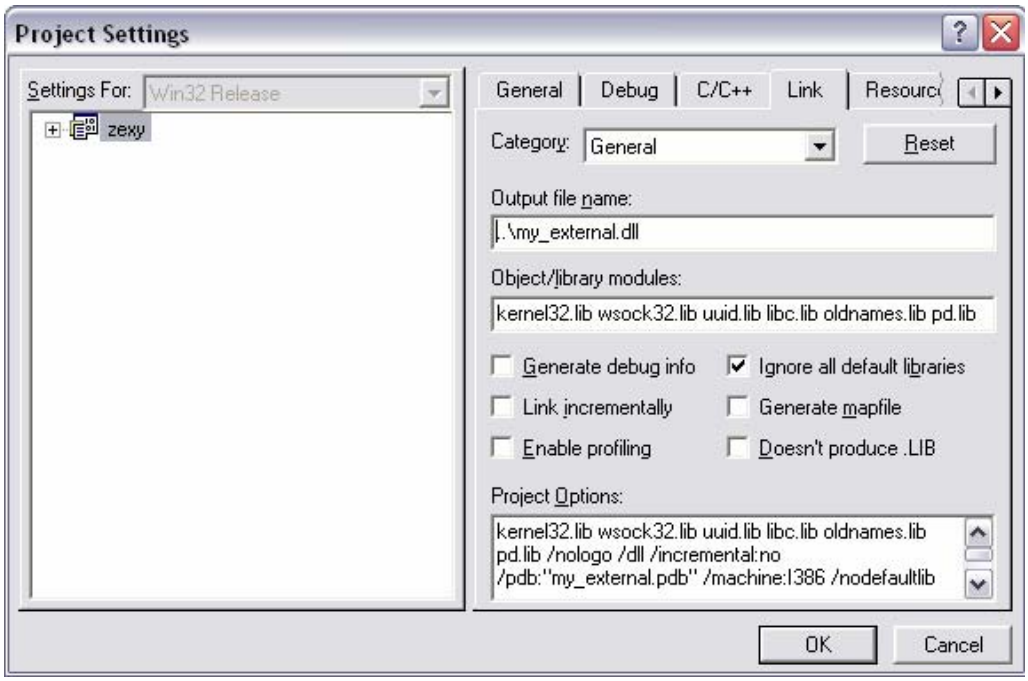
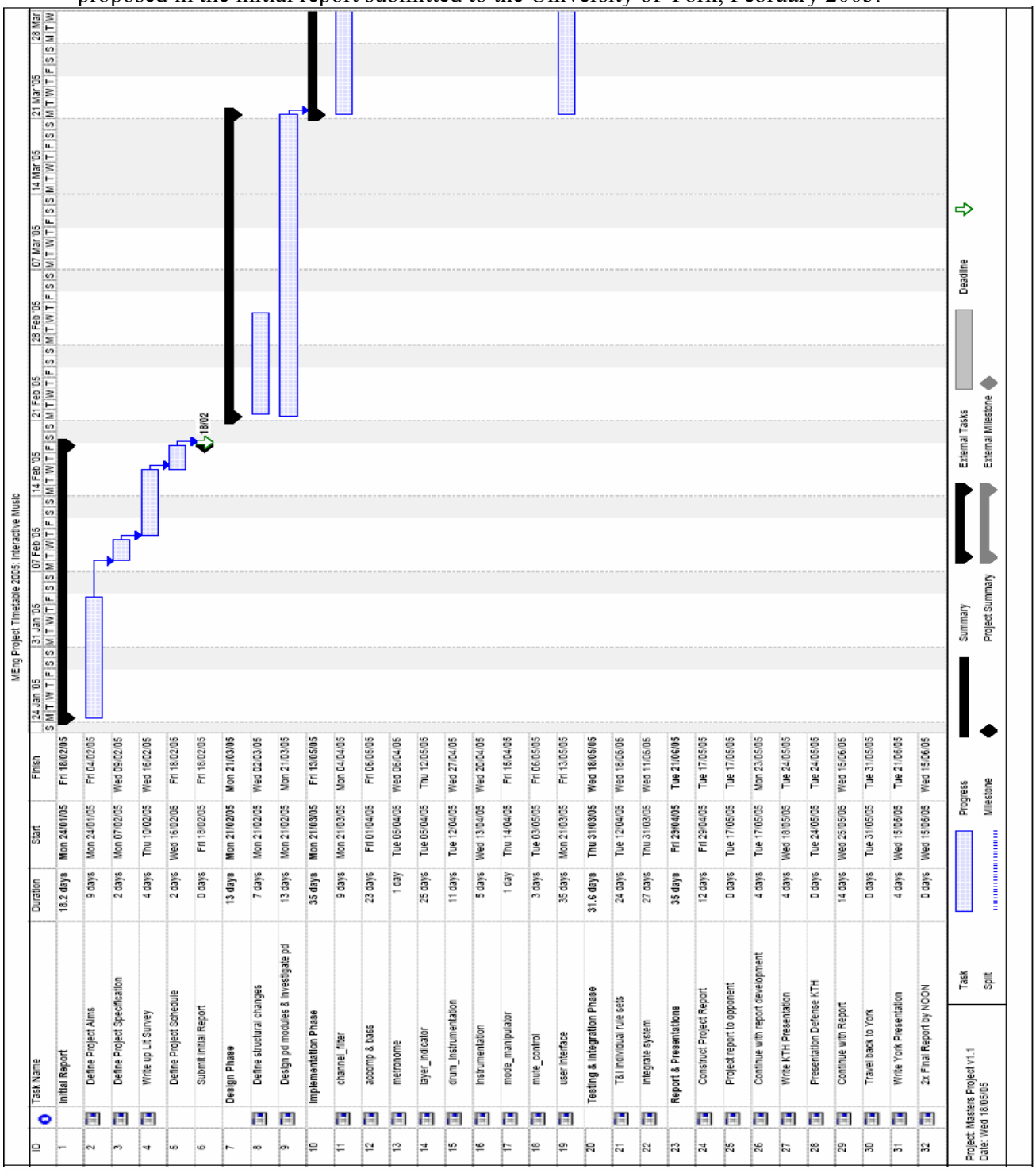


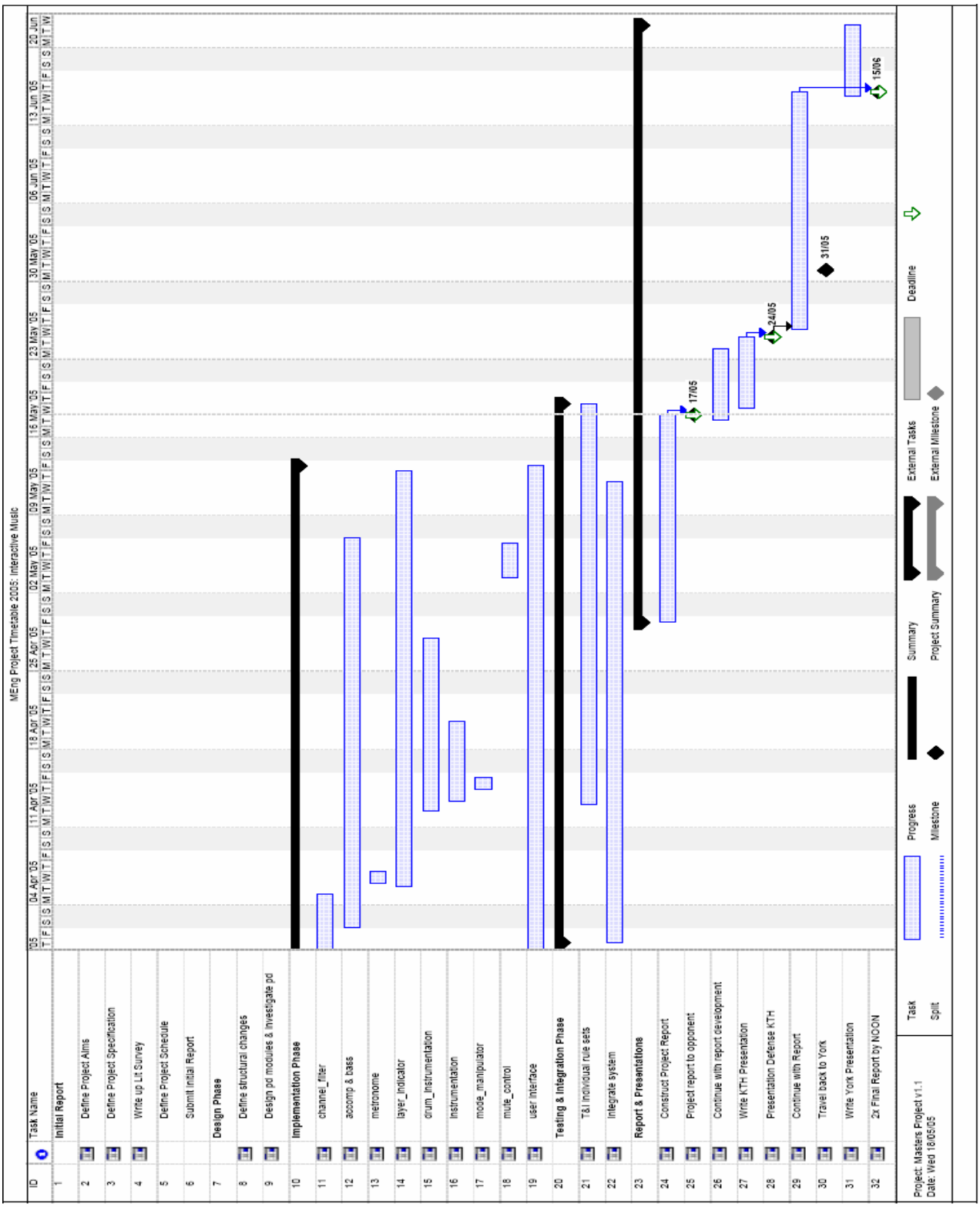
Figure E3: Link tab in project settings window

13. Place the cursor into the 'Project Options' text field at the bottom of the 'Link' window and move to the end of the text.
14. Rename '`export:zexy_setup`' to '`export:my_external_setup`'
15. Scroll the tabs to the right and click the 'Browse Info' tab.
16. Change the text field '`zexy.bsc`' to '`my_external.bsc`'. in the 'Browse info file name' text field.
17. Compile your external code using the build menu.
18. Build your 'my_external.dll' file using the build menu.
19. This file will appear in the zexy folder and needs to be copied into the extra folder of the pure data folder so that the new object can be recognised by `pd`.
20. Run `pd`.

F. Self Management & Gantt Chart

This Appendix details the final schedule of this project by means of a Gantt chart. Alterations in schedule can be found by comparing this chart with the original Gantt chart proposed in the initial report submitted to the University of York, February 2005.




MEng Project Timetable 2005: Interactive Music
ID
Task Name
Initial Report
Define Project Aims
Define Project Specification
Write up Lit Survey
Define Project Schedule
Submit initial Report
Design Phase
Define structural changes
Design pd modules & investigate pd
Implemantation Phase
channel_filter
accomp & bass
metronome
layer_indicator
drum_instrumentation
instrumentation
mode_manipulator
mute_control
user interface
Testing & Integration Phase
T&I individual rule sets
integrate system
Report & Presentations
Construct Project Report
Project report to opponent
Continue with report development
Write KTH Presentation
Presentation Defense KTH
Continue with Report
Travel back to York
Write York Presentation
2x Final Report by NOON
17/05
24/05
31/05
15/06
Project: Masters Project v1.1
Date: Wed 18/05/05
Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline

G. Score Format

```
FILETYPE          string    (Test_Score)

VERSION            int       (Score_Version_Number)

SCORENAME   string    (Song_Name)

METER             int int   (<mul> <div>)

TEMPO              int       (duration of quarter beat in uS)

KEY               string  string  (<G, F ...> <maj, min>)

PROGRAM           int int   (voice, channel)

BAR               int

CHORD       string, string, int  (root, type, dur)
(<G, F#.. > <maj, min, dom7,   maj7, dim... > <duration in mS>)

NOTE               int int int int   (pitch, Channel, velocity, dur)
```

H. Scores in extended pDM format

See electronic copy on CD

I. Notated Test Scores

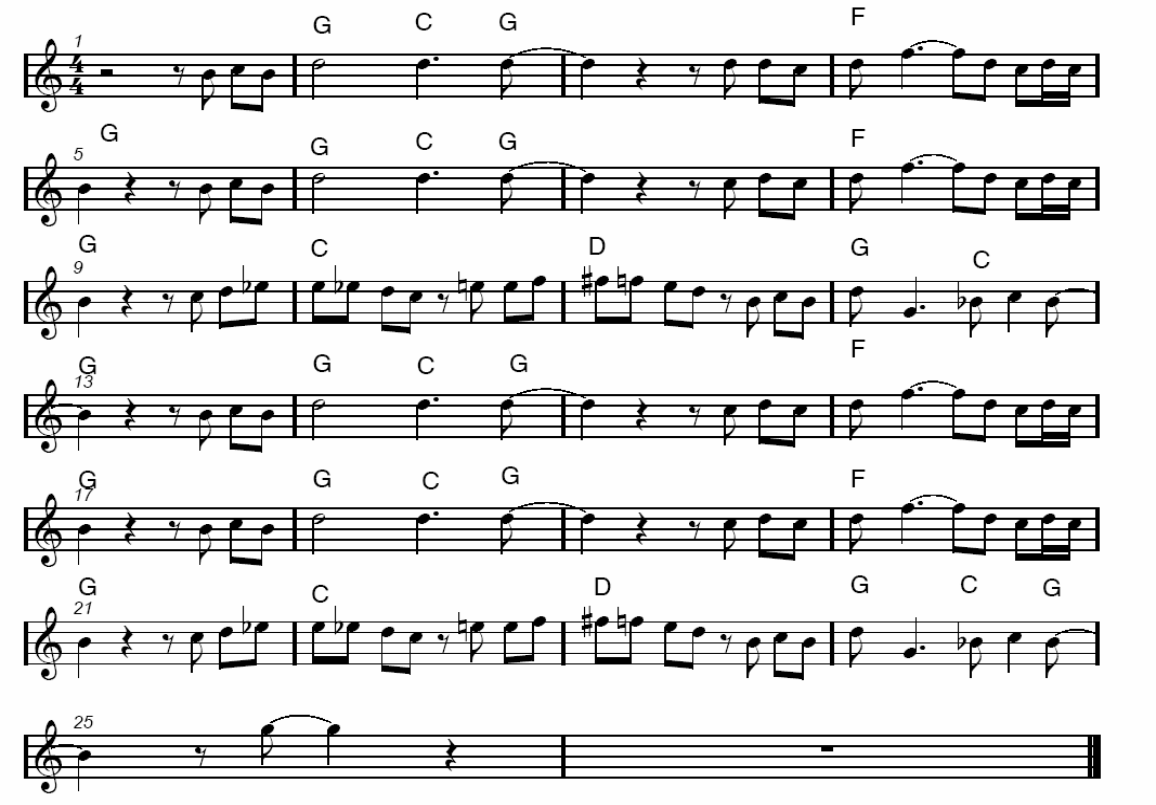


Figure I1: A Hard Days' Night

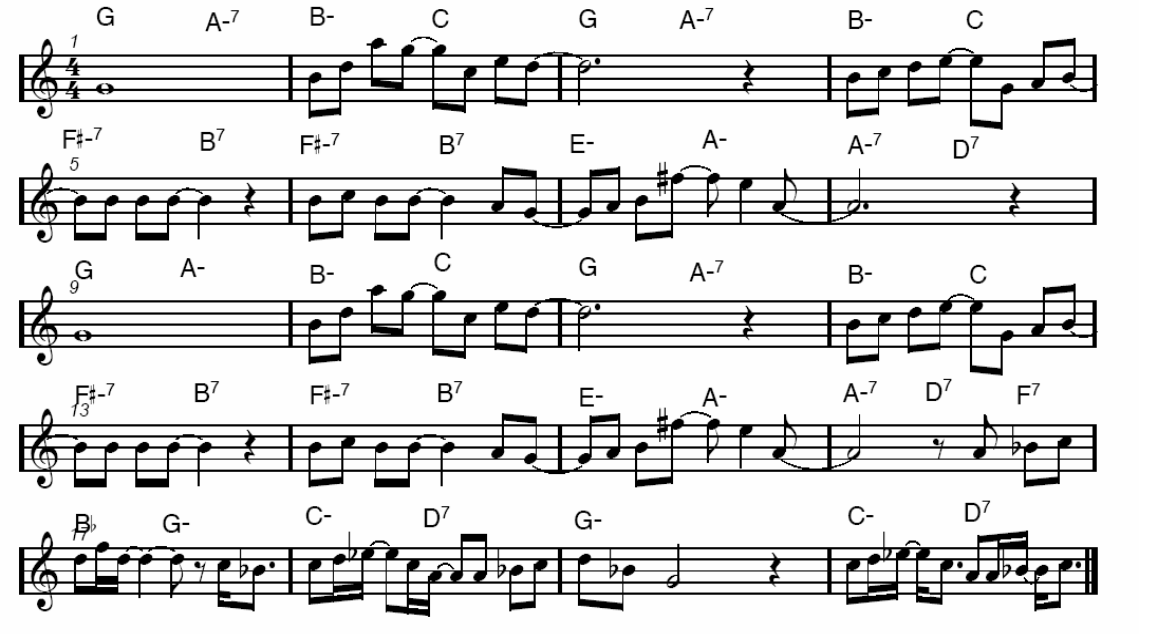


Figure I2: Here, There and Everywhere

J. Types of Cadence

a) Perfect
This type of cadence uses the *V – I* or *V – i* progression. Both triads are played in root position. The tonic note of the scale is sounded in the highest part. This type is the most decisive cadence and can be said to be conclusive analogous to a full stop at the end of a sentence.

b) Imperfect
The triads are not voiced in root position and /or the tonic is not the in highest part. This type is not as conclusive as a perfect cadence and can be analogous to a musical comma.

c) Plagal
The penult chord is *IV (iv)* moving to a final chord of *I (i)*. This technique can be used to end a musical phrase.

d) Deceptive
When a *V* resolves to a *vi*, it can sound deceptive or unexpected. Finishing on a I^7 also has a similar effect as it sounds as though it should be resolved a 5^{th} below.

e) Interrupted
This type is defined as when a *V* is resolved to a chord that bears no relation to the current tonic.

K. Introduction to Pure Data Concepts Used

This appendix is included to introduce the unfamiliar reader to the Pure Data Programming Environment and some of the common concepts used.

Building Blocks

pd is made up of basic core building blocks. The most commonly used blocks are illustrated in figure K1. An object block is created by typing a recognised and valid class name into the graphical box. Each object is written to carry out a specific function, the operation of which is usually indicated by the name of the object. Some object allows creation arguments to be specified in the same box too. The value or type of argument(s) can be typed into the object box after the name, separated by a white space.

The thicker black lines that appear at the top and bottom of blocks are inlets and outlets respectively. A block can be connected together using connectors. These connectors always start from the outlet of one building block to an inlet of another. This is the way in which data flows around a collection of blocks, called a patch.

Figure K1: Main building blocks in pd

A message box can contain single or a list of numerical values or symbols. Its contents can be sent to its single outlet by clicking on the box. See below for another method of outputting data from a message box.

A number box can store positive or negative integer/non-integer values. The contents of these boxes can be changed dynamically by the user when a patch is out of the edit mode.

Bang Object

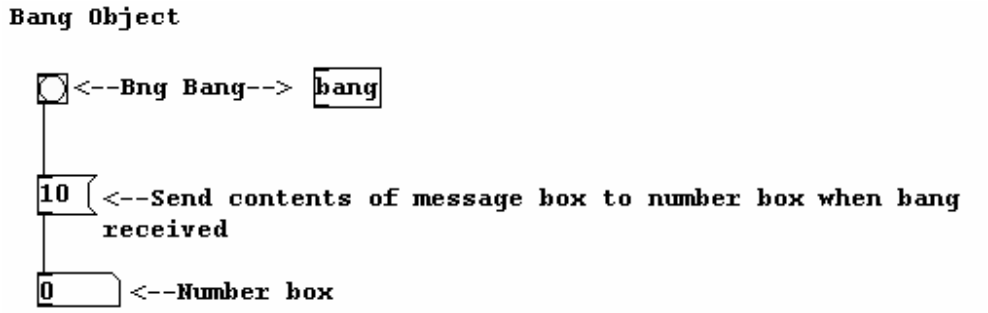


Figure K2: Bang object

The bang object forces each object to perform its specific operation when it receives a bang message. In figure K2, the message box outputs its stored data (the number 10) when it receives a bang message.

Toggle

The toggle object can be placed into a patch by navigating the menu in the patch window. It can be clicked to check the box to produce a '1' at its outlet or non-checked to produce a '0' at its outlet as shown in figure K3. The big_toggle object serves the same purpose. A bang message sent to the inlet of a toggle box changes the current status of the box to its opposite state

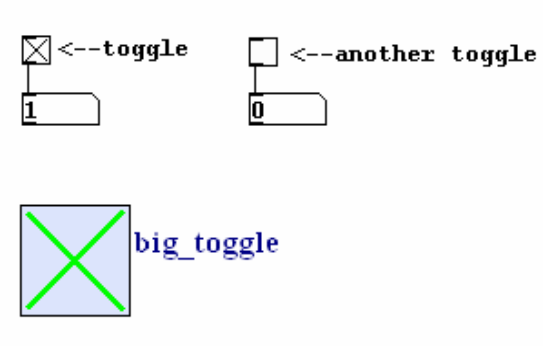


Figure K3: Toggle box

Mathematical Operations

Figure K4 shows how basic mathematical operations can be performed with `pd`. The lefthand side of the figure demonstrates a simple addition being performed. This also highlights the concept of hot and cold inlets. The leftmost inlet of the addition object is a 'hot' inlet. This means that any number present at this inlet will be instantly summed with the number at the rightmost inlet, and sent to the output. The right inlet is a 'cold' inlet. This means that any number present at this inlet will stay at the inlet until a new number arrives at the 'hot' inlet, where the two will then be summed and sent to the outlet.

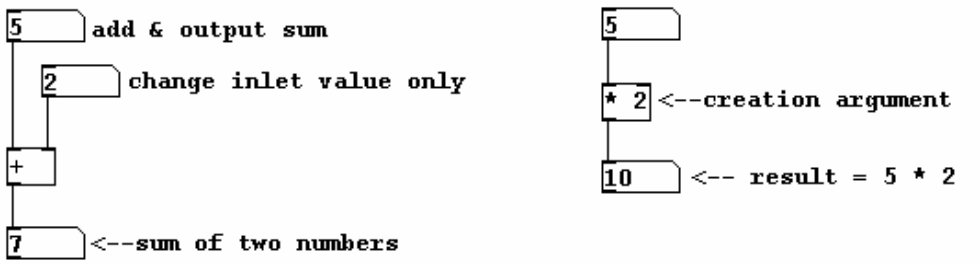


Figure K4: Mathematical Basics

The right-hand side of figure K4 illustrates the process of placing a creation argument inside the multiplication object.

Pack /Unpack Objects

The *pack* object in figure K5 shows how individual data values (atoms) can be combined into one list that is sent to the outlet of the object. In the most general operation of the *pack* object, it requires creation arguments matching the type and number of atoms being received. In `pd` f = float, s = symbol and b = bang.

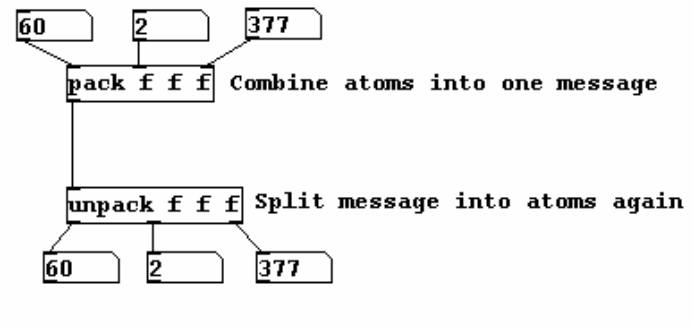


Figure K5: Operation of pack/unpack objects

The *unpack* object serves the opposite purpose and splits an incoming list of data into individual atoms.

Send/Receive Objects

An important concept in `pd` comes into use with the *send* and *receive* objects illustrated in figure K6. Any type of data can be sent from a *send* object to a *receive* object. The creation argument in the *receive* object must match the one in the *send* object if the data is to be passed to the correct object.

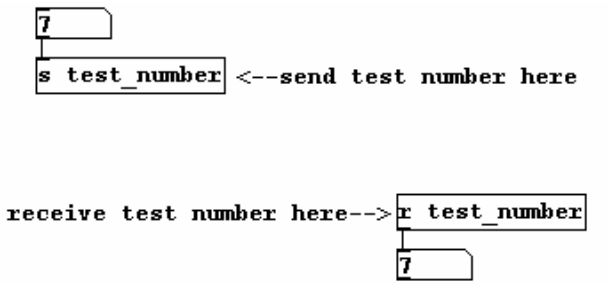


Figure K6: Demonstration of send and receive objects

Trigger Object

Trigger outputs its input from right to left and converts to the types indicated by its creation arguments. Figure K7 shows how this object is useful for splitting data into multiple paths. The same effect can be achieved by connecting many connectors from one outlet, but this can introduce problems with timing of data distribution.

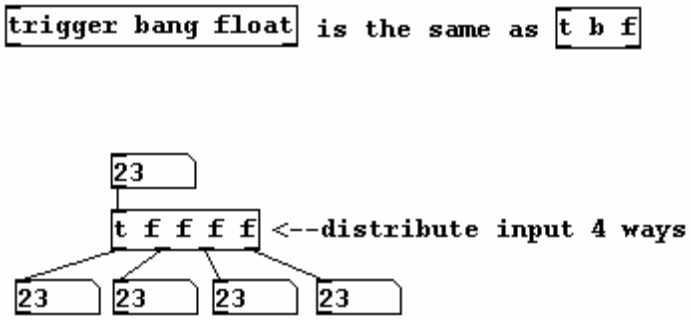


Figure K7: Use of the *trigger* object to distribute data

Spigot Object

This object passes the data present at the left inlet if the control value at the right inlet is positive, and blocks the incoming data if the control value is non-positive.

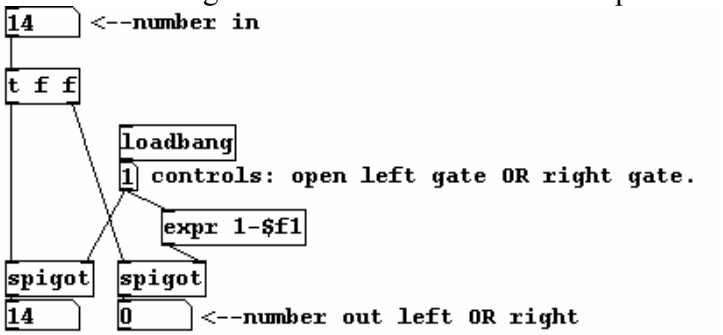


Figure K8: Implementation of a pass/block gate with the *spigot* object

This object can be used to create a function that can either route a data flow to one object or another as shown in figure K8.

Select Object

A very useful object that analyses the incoming data to see if it matches any of the creation arguments specified. If a match is found with the creation argument, a bang message is sent to the outlet that is the same number of places from the leftmost outlet as the creation argument. If no match is found, a bang message is sent to the rightmost outlet.

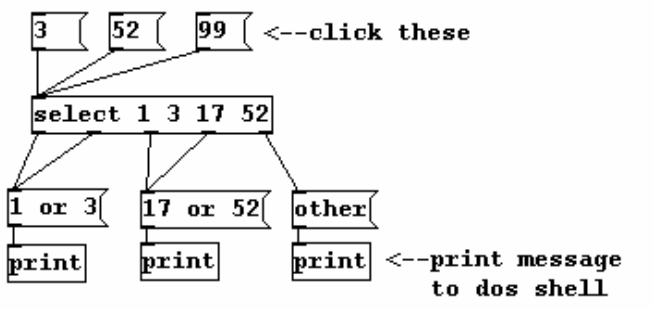


Figure K9: Using the select object to split and route incoming data flows

Figure K9 shows how a *select* object is used to identify that data specified in the creation argument is present at the inlet and trigger a message containing that data. This is useful, as data is not passed though the object, just a bang to show that a match has been found.

Sub patches

Another important concept in `pd` is that of sub-patch (or more specifically, sub-patches). A separate function can be created in its' own patch window and have data sent to it for processing from another patch window via inlets and outlets located in the sub-patch.

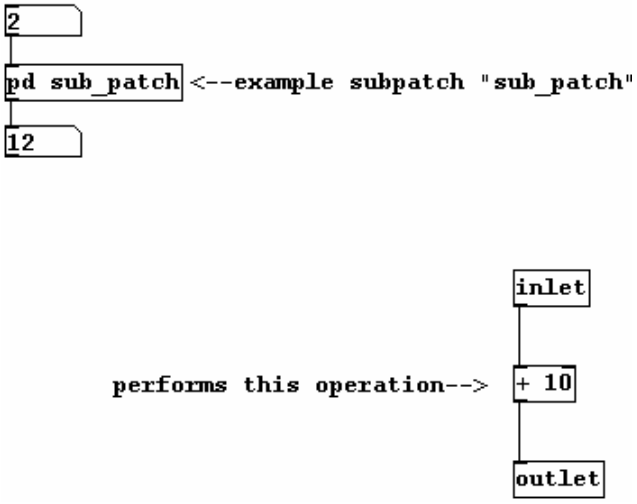


Figure K10: Demonstration of a function contained in a sub-patch

Figure K10 shows how the sub-patch called "sub_patch" contains the function to add 10 to the input number and then send the result to its outlet.

L. Pure Data Code

See electronic version on CD

M. Full Track Listing for the Accompanying CD-ROM

An accompanying CD-ROM contains the following sound examples of the system in use. Please see the **index.html** file to navigate the examples.

Here, There and Everywhere

1. No user movement
2. User in Happy position (1,1)
3. User in Tender position (-1,1)
4. User in Sad position (-1,-1)
5. User in Angry position (1,-1)
6. User moving in circular motion as illustrated in figure M1
7. User moving in cross motion as illustrated in figure M2

A Hard Days Night

8. No user movement
9. User in Happy position (1,1)
10. User in Tender position (-1,1)
11. User in Sad position (-1,-1)
12. User in Angry position (1,-1)
13. User moving in circular motion as illustrated in figure M1
14. User moving in cross motion as illustrated in figure M2

System integrated with pDM

15. User moving in circular motion as illustrated in figure M1
16. User moving in cross motion as illustrated in figure M2

Two video examples of system being tested by two individuals are also given on the CD-ROM

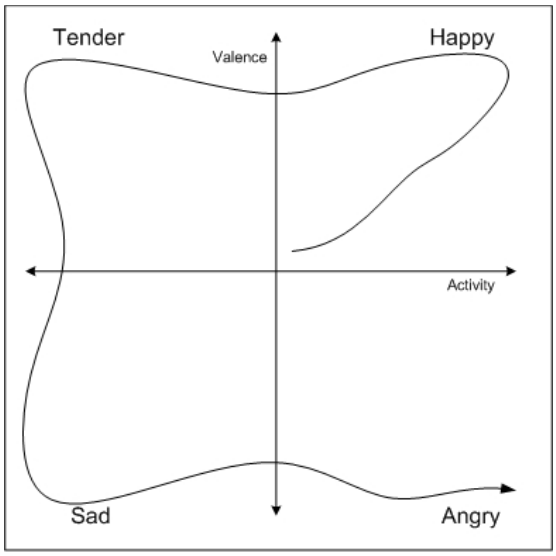


Figure M1: Circular user motion in Activity-Valence control space

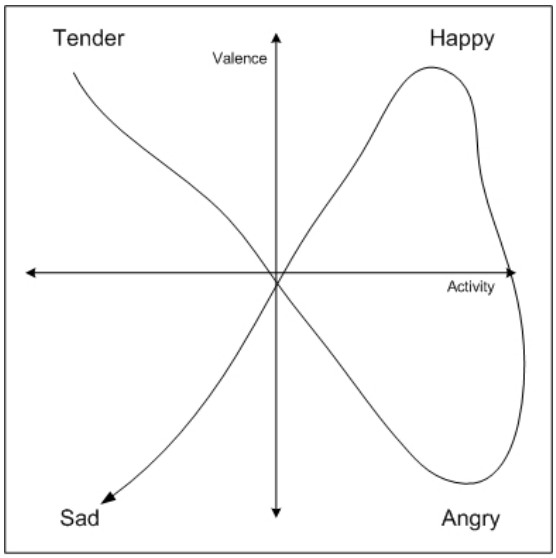
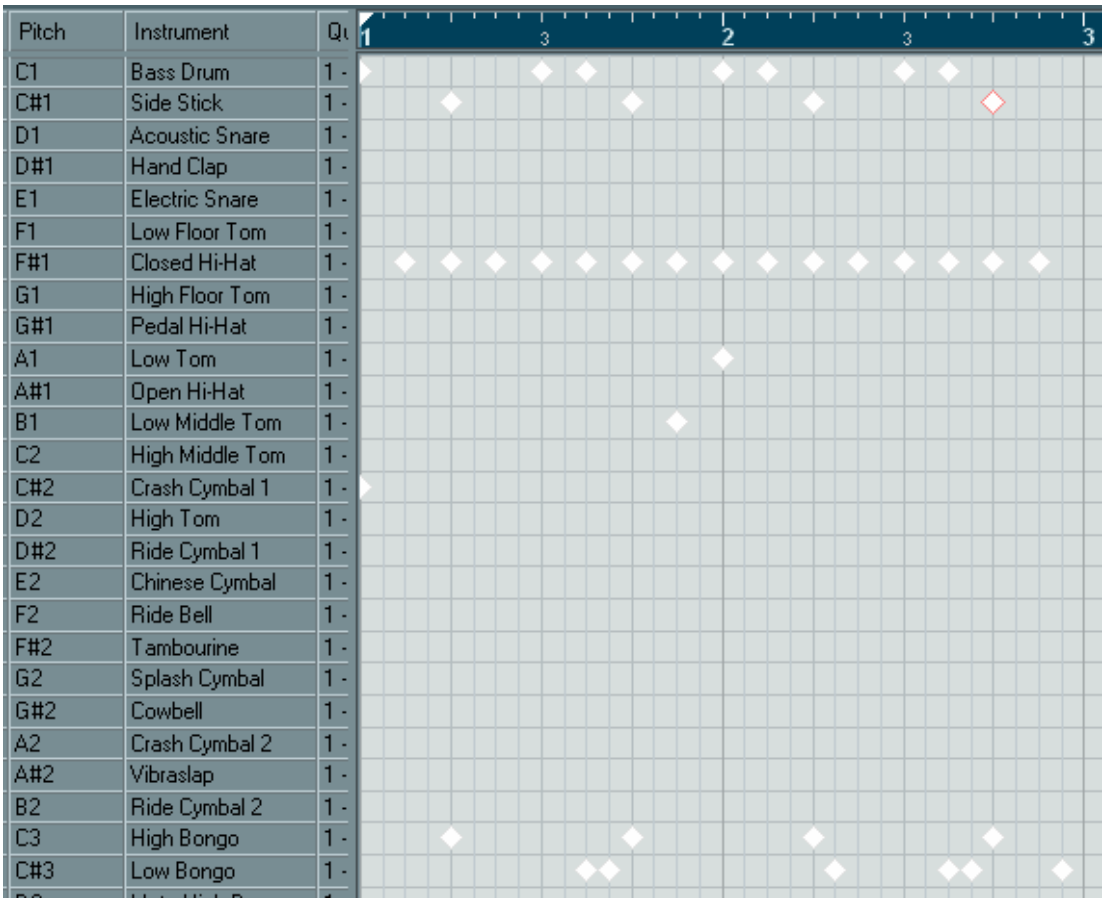


Figure M2: User moving in cross motion in Activity-Valence control space

N. System Testing Results

Original 2-bar drum track in MIDI drum editor format



FigureN1: Original drum track in MIDI drum track format

Test Number: 1
User position in Activity-Valence control space: (1,1)
Mapped emotion: Happy
Chord Progression: C Major 7 to D Major 7
Key: C Major

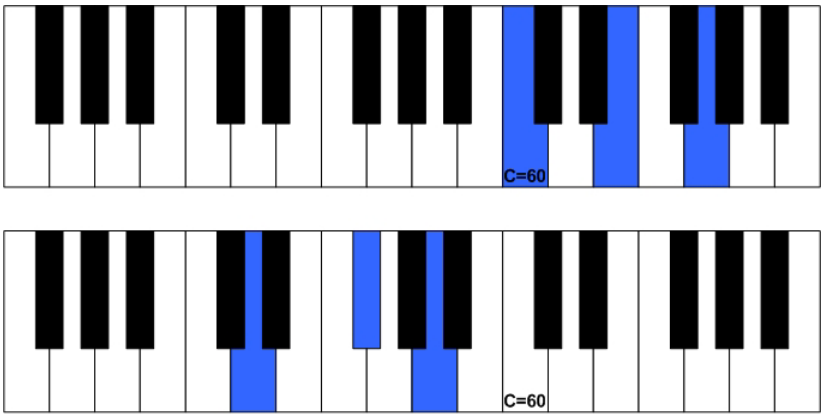


Figure N2: Accompaniment result for given chord progression when user is at position (1,1)

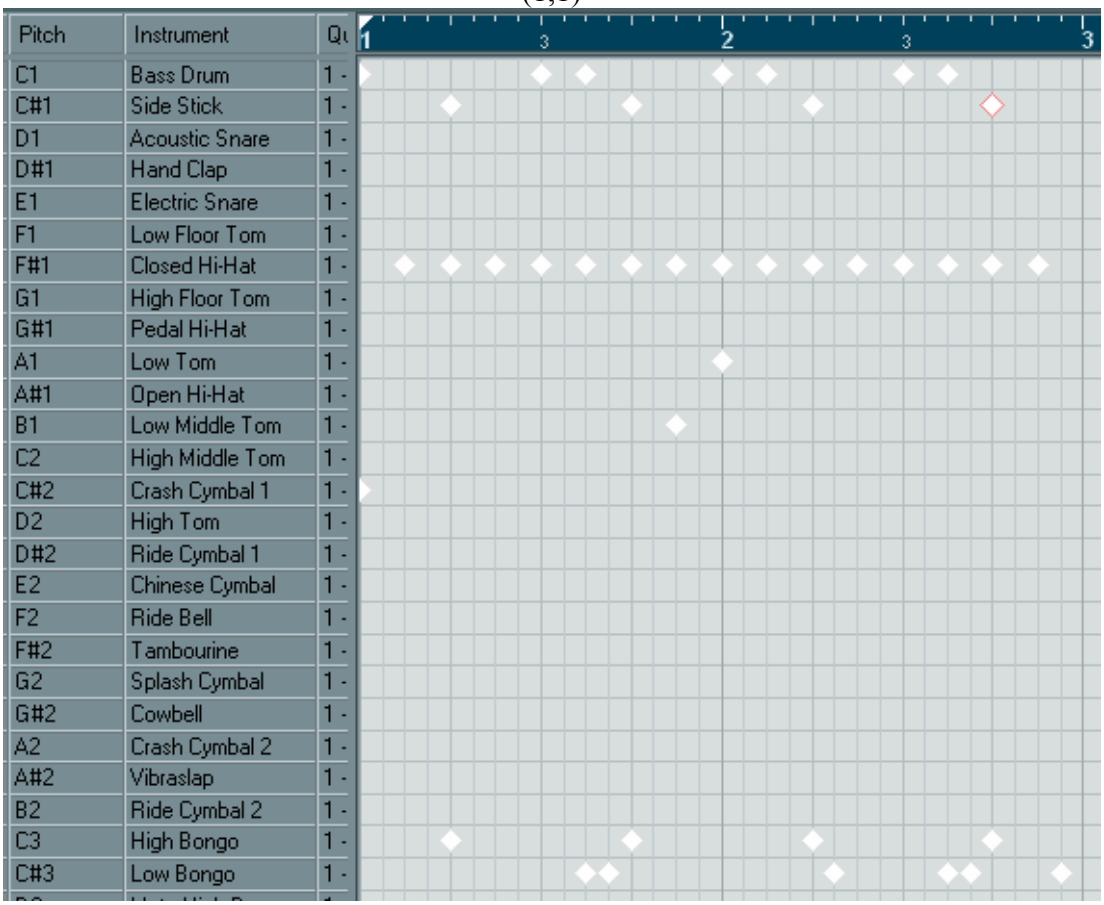


Figure N3: Drum data processed by system when user is in position (1,1)

Test Number:2
User position in Activity-Valence control space: (-1,1)
Mapped emotion: Tender
Chord Progression: C Major 7 to D Major 7
Key: C Major

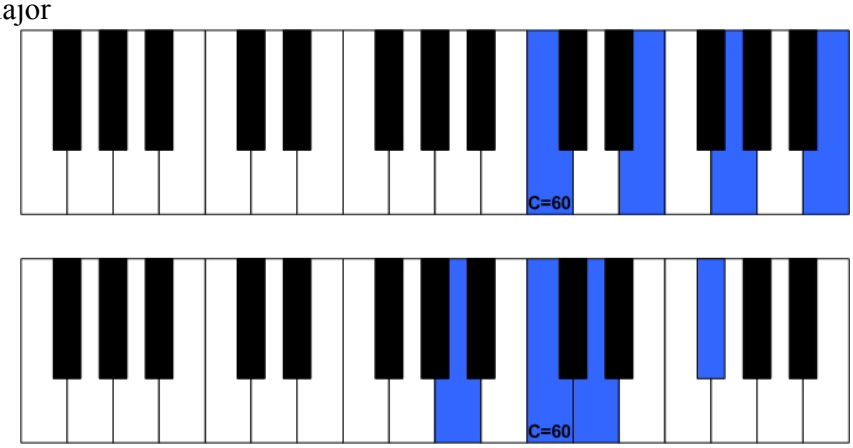


Figure N4: Accompaniment result for given chord progression when user is at position (-1,1)

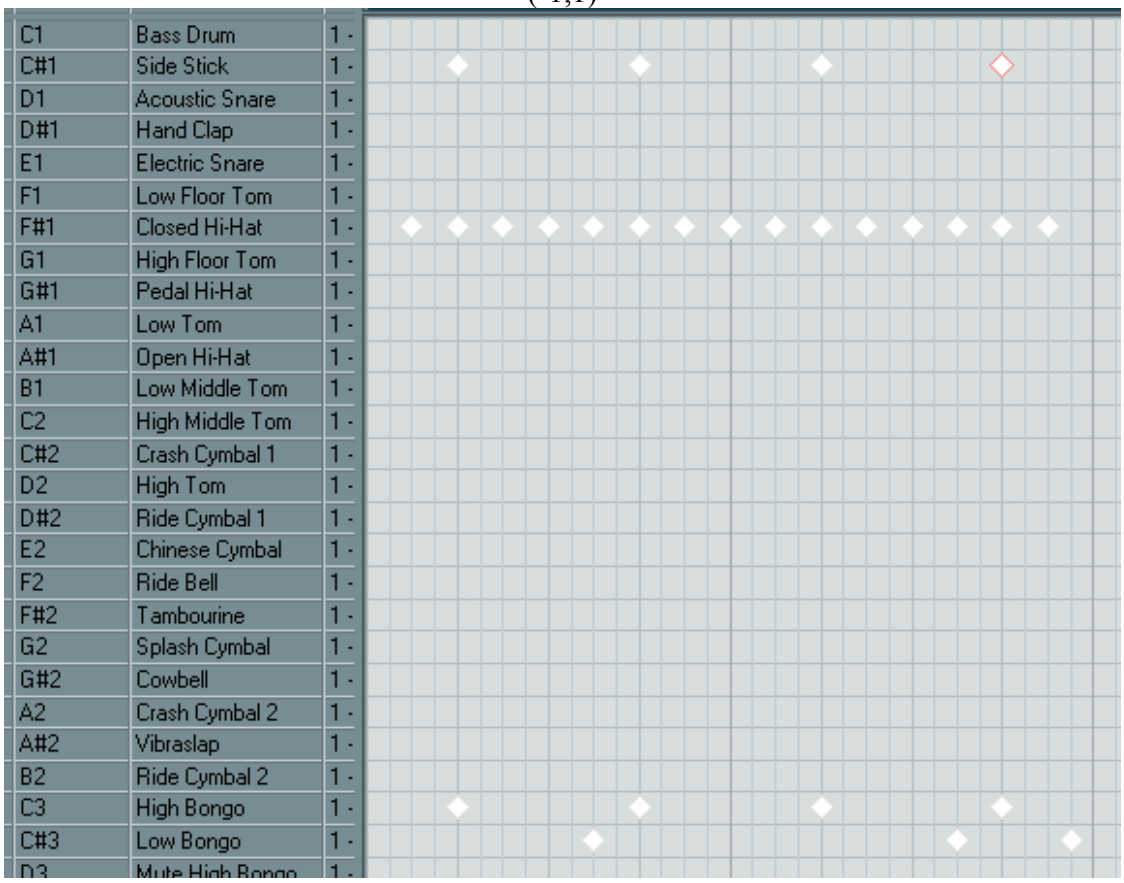


Figure N5: Drum data processed by system when user is in position (-1,1)

Test Number:3
User position in Activity-Valence control space: (-1,-1)
Mapped emotion: Sad
Chord Progression: C Major 7 to D Major 7
Key: C Major
Minor Mode: Aeolian

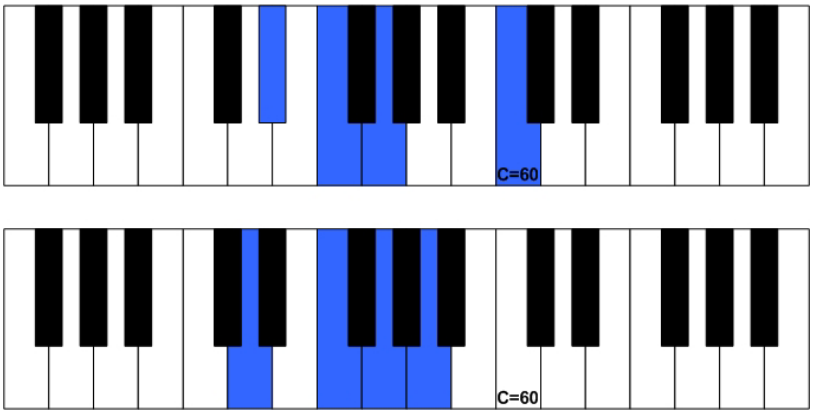


Figure N6: Accompaniment result for given chord progression when user is at position (-1,-1)

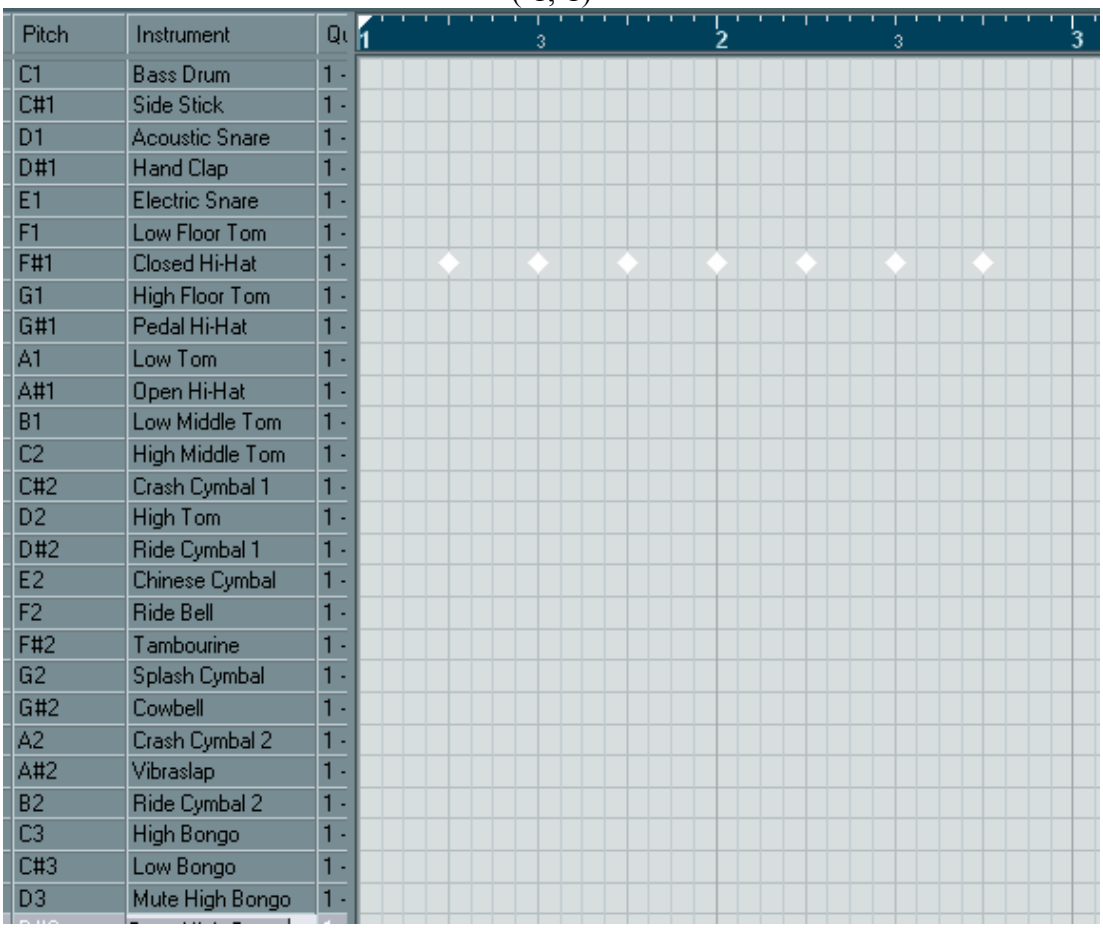


Figure N7: Drum data processed by system when user is in position (-1,-1)

Test Number: 4
User position in Activity-Valence control space: (1,-1)
Mapped emotion: Angry
Chord Progression: C Major 7 to D Major 7
Key: C Major
Minor Mode: Aeolian

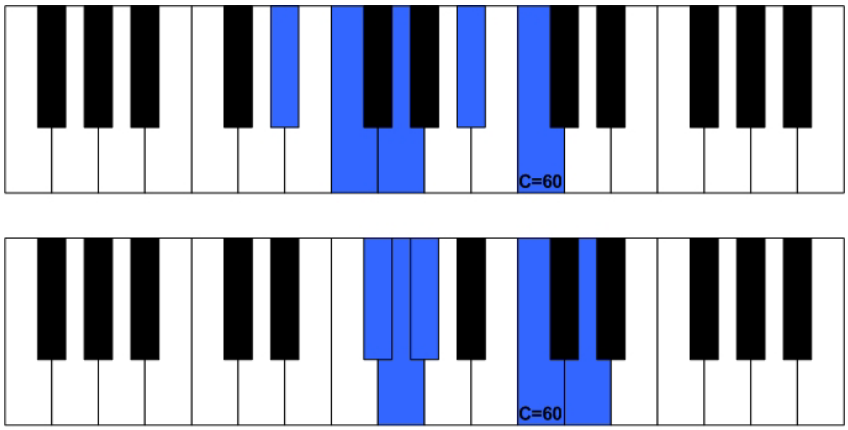


Figure N8: Accompaniment result for given chord progression when user is at position (1,-1)

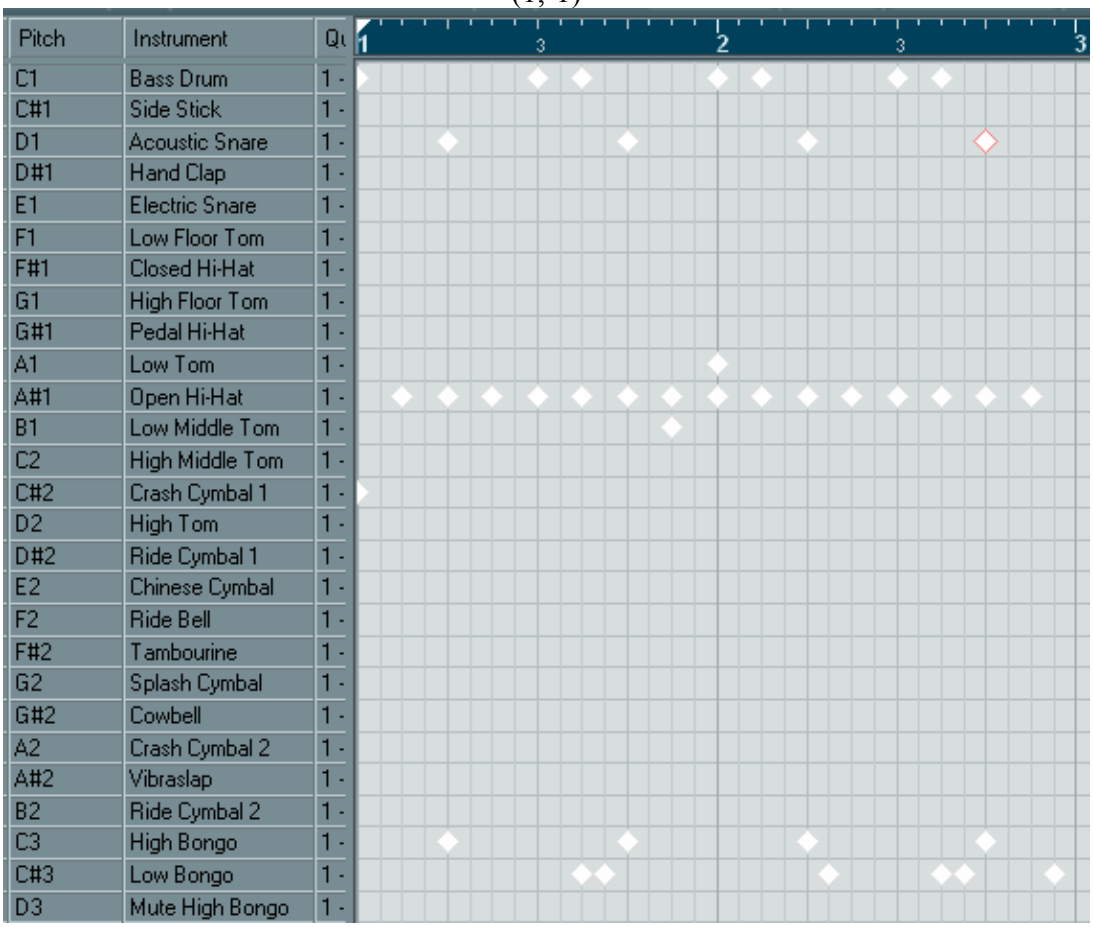


Figure N9: Drum data processed by system when user is in position (1,-1)

The above test results confirm that the system is performing as specified in each of the co-ordinates set out in the objectives in section 3. The voicing of the accompaniment changes to reflect the properties of the harmony identified in figure 7. The rhythm changes as shown in the MIDI drum map window. Layers 1 and 2 are toggled on and off according to user input to remove beats. Drum instrument groups are also removed as well as changed to reflect the input emotion.