Sonic Art: Recipes and Reasonings

Adrian Moore  David Moore
The University of Sheffield  The University of Sheffield
Department of Music  Department of Music

December 2011
## CONTENTS

2.2.4 Filters .................................................. 39
2.2.5 Granulation ............................................. 43
2.2.6 Pitch and Frequency shifting ......................... 44
2.2.7 Spatialisation, Delay based effects ................. 45
2.2.8 Synthesis ............................................... 48
2.2.9 Cross-Synthesis ........................................ 49
2.3 Compound tools .......................................... 50
2.4 Csound and Blue .......................................... 53
  2.4.1 Csound ................................................. 54
  2.4.2 Blue ................................................... 54
  2.4.3 Blue examples: A soundfile player .................. 54
  2.4.4 Blue examples: A brassage filter machine ......... 55
  2.4.5 Blue examples: A more complicated Python example:
      Multiple soundfile brassage .......................... 57
  2.4.6 Blue examples: A special granulator ............... 57
  2.4.7 Blue examples: Comb Filters ......................... 59
  2.4.8 Blue examples: Convolution: a different kind of cross
      synthesis .............................................. 60
  2.4.9 Blue examples: The Phase Vocoder - time stretching .. 61
  2.4.10 Blue examples: The Phase Vocoder and frequency shift-
      ing/stretching/blurring ................................ 62
  2.4.11 Blue examples: Spectral Warping ................... 63
  2.4.12 Some final thoughts on Csound ..................... 64
2.5 SuperCollider .............................................. 64
2.6 The three amigos: SuperCollider, Csound, Puredata .... 64
2.7 SoX .......................................................... 67
2.8 Reverberation .............................................. 68
2.9 Mixing ...................................................... 68
  2.9.1 Setting up the reverberation channel ............... 68
  2.9.2 Mixing techniques ..................................... 69
  2.9.3 When to start mixing .................................. 70
  2.9.4 Where to start mixing .................................. 70
  2.9.5 Cataloguing soundfiles ............................... 73
3 The theory of opposites ......................................... 75
  3.1 Introduction .............................................. 75
    3.1.1 Questioning, Opposing, Arguing ..................... 75
  3.2 Effects and descriptors ................................... 78
    3.2.1 The theory of opposites ............................ 81
  3.3 Structure .................................................. 81
    3.3.1 Smalley’s use of structure .......................... 81
    3.3.2 Other structure(s) ................................... 82
    3.3.3 Beginning, Middle, End .............................. 83
  3.4 Form ....................................................... 83
    3.4.1 Does this matter? .................................... 84
  3.5 Originality ................................................ 84
## CONTENTS

3.6 A very obvious conclusion ........................................ 85

A USSS toolkits ....................................................... 87
A.1 A1. Downloading and installing software ..................... 87

B Composer recollections ............................................ 89
B.1 Adrian Moore .................................................... 89
B.2 Alistair MacDonald ............................................... 90
B.3 Andrew Lewis ................................................... 90
  B.3.1 First I find, then I seek .................................... 90
  B.3.2 The trash can is your friend ............................... 91
  B.3.3 Listening .................................................. 91
  B.3.4 Time ...................................................... 91

C Binary Representations ........................................... 93
C.1 Decimal (base 10) ................................................ 93
C.2 Binary (base 2) .................................................. 94
C.3 Counting in binary .............................................. 94
C.4 Bits, Bytes and leading zeros .................................. 95
C.5 Adding binary numbers ......................................... 95
C.6 Representing negative numbers ................................. 95
C.7 Fractional numbers ............................................. 95
C.8 Floating point .................................................. 96
C.9 ASCII Strings ................................................... 96
C.10 Hexadecimal .................................................... 96

D Puredata (Pd) ....................................................... 99
D.1 Basic units of programming Pd ................................. 100
  D.1.1 Data types ................................................ 101
  D.1.2 Numbers and range calculations ......................... 101
  D.1.3 Displaying values and messages using print .......... 102
  D.1.4 Range mapping .......................................... 103
  D.1.5 Getting numerical data from MIDI controllers and keys 104
  D.1.6 Lists ..................................................... 105
  D.1.7 Sending and receiving data without connections ........ 106
  D.1.8 Tips for exploring Pd .................................. 107
D.2 Signal Processing in Pd ......................................... 108
  D.2.1 The audio signal data type ............................... 108
  D.2.2 Simple signal manipulation .............................. 109
  D.2.3 Audio objects for generating signals .................... 109
  D.2.4 Adjusting the range of signals ........................... 110
  D.2.5 Other simple manipulations on signals .................. 110
D.3 Sampling ........................................................ 112
  D.3.1 Why do we need both types? ............................. 112
  D.3.2 Buffer objects: table and array ......................... 112
E  Audio driver technology: ALSA, Jack etc. 113
0.1 Preface

This book was originally written for undergraduate and postgraduate students specialising in computer music composition and sonic art at The University of Sheffield Department of Music. Whilst the tools mentioned here are part and parcel of our teaching program, they are very flexible and easy to use; there seemed no reason not to make both the tools and the book available to all.

However, please note that this is not a book that ‘starts at the very beginning’. Indeed it assumes quite a bit both about computer operation and musical aesthetics.

Thanks go to my colleague Dave for his technical expertise, Martin Curtis-Powell for his careful and attentive reading of the text and to all the composers that have contributed personal recollections over their extensive careers. Thanks also to the University of Sheffield Faculty of Arts for assisting us with this project.

Adrian Moore / Dave Moore.
Chapter 1

What is sound?

Student: What sounds can I use in my composition?
Teacher: All sounds are available to you.
Student: Okay, so where do I start?
Teacher: Start with a sound that interests you.

1.1 Introduction

1.1.1 Jumping in at the deep end

This book is about composing with sounds. Composition can, but does not have to start with sound but sound is all around us and quite often, we may wish to have more control over it than it has over us. The prevalence of recording technology means that we can capture sound quite easily and store it on the computer. But what do we do with it then? Trevor Wishart opens his book Audible Design (Wishart, 1994) by making three assumptions:

1. Any sound whatsoever may be the starting material for a musical composition.
2. The ways in which this sound may be transformed are limited only by the imagination of the composer.
3. Musical structure depends on establishing audible relationships amongst sound materials.

Each assumption opens up a wealth of opportunity. Our book hopes to tie down some of the possibilities, block off some of the dead-ends of sound manipulation and suggest a number of meaningful methodologies for selecting, developing and mixing sound so that as a composer, you can make something musical. Wishart mentions musical composition but for many, the experience of shaping sounds will have little to do with the pitches and rhythms of Western Classical or Popular music. That is not to say that the music we all listen to
CHAPTER 1. WHAT IS SOUND?

will not have an effect upon us and will not influence how we work with sound in the raw: it is just that if we try to make a ‘musical composition’ of pitches and rhythms with raw recorded sounds, chances are, it will sound terrible.

So let us throw ‘music’ out of the window but bear in mind ‘musical’: something with phrases, starts, ends, middles, highs, lows, louds, softs, breaths, pace, shape, form, emotion and energy. Think about working with sound like a potter works with clay. Take a sound and mould it to the shape you want. On some occasions that may be a fixed shape; at other times, we may end up with something completely random and fluid. What is it then? Clay, yes, but how do we describe the shape? As soon as we start describing things in terms of abstract details (thin, curved, round) and making references to other objects and emotions that have personal relationships with us (lively, frightening) the sooner we get a grip on how we can develop a structure.

As this book progresses we will look at how we might define a sound. It is unfortunate that in a book we have to define sounds with words but the flip side here is that the process is quite liberating as it forces analysis.\footnote{Similar too is the process of making a graphic score of an existing piece: it focuses our listening as we move from sound to graphic.} We will listen to sounds and consider developing them over time. Time plays a huge role in the structure of any piece of sonic art. When we come to mixing sounds (one after the other, one on top of the other) we are shaping sound in time: sound is not just heard as A to B but felt over a duration.

Having thought about describing sound, we will look at techniques to develop sound, techniques that work well with the salient properties that are contained within our sounds. We will look at a number of different software packages and concrete, usable examples will be given. All the software used will be open source and therefore free to install and run on almost any computer.

Finally we will think about how we might tie sounds and techniques together and how we can build up a fluid set of skills that enable us to quickly hear the acousmatic-potential of a sound and develop it into something new. We suggest that it might be possible to work ‘against’ the sound, reacting for the most part in a negative fashion, always being resistive to what a sound wants to do. For example, if a sound is noisy, chances are, over time it might want to become less noisy. (It is certainly easier for it to become less noisy than more noisy though there are no hard and fast rules here). So we will outline a principle of defining sounds across poles (light/dark, soft/hard, high/low, fast/slow) and suggest that if a sound is slow, you may want to consider techniques that speed it up etc.

So if you want to get cracking and really jump in at the deep end, chapter 2 might be a good place to start. If you want to see how techniques and sounds interact, chapter 3 will help. This book is meant to be both a user-guide and a text book. What it is not is a piece of scholarly literature philosophising on the nature of sonic art. It is vital to note that if you do jump in at the deep end, the life preservers and water-wings are the texts and compositions of the past. They are not there just to fill shelves or disk drives; they are there to be
read and listened to. Listening and reading to as much tangential music and literature will help make your journey as a composer a faster, more exciting and pleasurable ride.

1.1.2 Context of the book

Given the number of composers manipulating sound to all sorts of ends, and given the huge array of software tools on offer it is hardly surprising that many have put some of their thoughts on paper. For academic readings concerning the relationship between theory and practice, John Young (Young, 2007) sums up very effectively the quite personal relationship a composer has with his/her sounds. And he mentions a small number of composers we consider to be highly influential in sonic art theory and practice, composers that have taken the time to write about what they do: Trevor Wishart (1986; 1994; 1996), Denis Smalley (1986; 1996; 1997; 2007), Simon Emmerson (1986; 2007) and Jonty Harrison (1998; 1999). It may also be useful to reflect on recent publications that deal with sound design (Farnell, 2010). You may find quite a lot of terminology drawn from Denis Smalley’s academic writings (words such as gesture, texture, utterance, energy, motion, space, environment), not because this book needs to be bound in academic terminology but simply because they are the right words for the job at hand.

Additionally we can see numerous books that deal directly (or in passing) with the techniques of sound manipulation, synthesis and sound design. Miller Puckette’s (2007) software Puredata(Pd) is one of the key resources we have used to make toolkits for sound manipulation. A more focused introduction to this software can be found in Johannes Kriedler’s book on the same software (2009). Having access to a book is great: fortunately both the above texts are available as free downloadable pdf documents. We have constructed a number of toolkits that you can use and develop. Our toolkits exist in Pure Data and Csound (using an excellent interface designed by Stephen Yi called Blue (Yi, 2008)). Links for downloads of all software are available from the University of Sheffield Sound Studios web pages http://www.shef.ac.uk/usss. We will also use a small amount of Supercollider code and a very timely book on Supercollider (Scott Wilson, 2011) has just appeared. Intimate knowledge of software tools over the years allows you to be flexible when designing sound manipulations. However, this book only assumes a ‘working’ knowledge of the computer (how to get sound in and out, how to edit a stereo file and perhaps how to mix two files together). The computer is your instrument and it is important to practise its scales and arpeggios in order to gain proficiency. A very recent primer in Pure Data comes in the form of Andy Farnell’s excellent book, Designing Sound (Farnell, 2010) and his code examples are available from the MIT press website.

It is important to say a word about right and wrong in sonic art composition. Although the composer is in control, unless you want to play your music only to yourself, you should consider your audience. It is important to listen to your music as others might hear it otherwise you tend towards self-affirmation (you know what is about to happen so when it does, you are satisfied). Play your
music to others during the course of your composition and be prepared for frank comments. Normally what feels right is right but most importantly, you need to know why it is right. It may well be easier to start off working out what is wrong with the music that you are listening to. (Medical discovery works in this way for the most part - reaction (to illness) rather than anticipation and precaution). This book arises from countless hand-outs and wiki pages supporting courses in electroacoustic music composition at The University of Sheffield. Students start ‘messing around with sound’ in year 1 with only their previous school studies to support them. Whether they can identify the right and wrong by the end of year three is irrelevant. Their final year works are living proof that something has changed, decisions have been taken, understandings made. We hope to speed this process further still by suggesting some interesting ‘recipes’ for sound manipulation. To back up our suggestions we will provide a number of ‘reasonings’ based upon the authors’ experience. Like any good cook book, eventually you won’t need to use it but knowing where you can lay your hands on it may well be reassuring.

1.2 Composition and Recording

Acousmatic composition begins with source material that is processed and reflected upon. The acousmatic stance asks that you listen to your sound in terms of raw qualities of energy, direction, colour, density and texture (amongst other attributes). We need to come up with a simple, easy to use language to help ‘tag’ our sounds (or portions of them). Although identification of the sound in terms of more personal qualities or emotions, or indeed the identification of a real-world sound may be ideal, try to dissect the sound in front of you into component parts.

If composition begins with sound, we need to record that sound. It may be recorded with its environment fully intact (a passing car, voices in the street, an airport tannoy) or it may be an object taken from its environment and recorded in the silence of the studio. When this happens you can normally get the microphones quite close to the source and use them like a microscope on a visual object, examining from multiple angles. At this early stage it is vital to get the recording you want at the highest quality possible. Bad sound in equals bad sound out, no matter how good your composition process might be.

1.2.1 Recording

We normally record in stereo and if we were outside we would aim for a good stereo field, to enable us to capture a wide variety of sources from left to right. In the studio our left and right need not be close together emulating the head. If we were recording some marbles in a bowl (and were moving the bowl around) we might place one microphone at the very left edge of the play space and the other at the very right.

As the marbles spin/swirl from the left to the right as we gently rotate and
undulate the bowl, the resulting sound will move from the left ear / speaker to the right ear / speaker. Already we have made a major, non-real-world transformation and potentially created a texture with a rolling feel and a sense of acceleration from left to right. Quite a lot of composition can be achieved during a playful recording process.

Other important factors to remember when recording.

- **Dynamic range.** Sounds that are just too quiet are never going to be any use. Sounds that vary from loud to soft give you plenty of room to select. When recording ensure that you get a good signal level on the recording device otherwise any future amplification will just bring up the level of background noise. This is called signal to noise ratio.

- **Texture/Gesture.** You may well want to record sustained sounds and discrete sounds. Take the previous example. Whilst having lots of marbles swirling around is an excellent texture, one marble moving once from left to right might be an example of a simple ‘swipe’ gesture.

- **Focus.** Bringing a sound close to the microphones gives a sense of intimacy and normally aids in increasing the dynamic range. Moving the sound away from the microphones narrows the stereo image.

So already, during the recording process we have identified a number of spatial dimensions within which to play.

- **Left/Right.** Perhaps the most obvious. The source may move (marbles) but if it does not (your voice), perhaps you should change your focus and sing towards one microphone then the other. It is often more natural to move in accordance with your vocal utterance (though it does not look that graceful in the studio) than panning a mono signal in a Digital Audio Workstation or DAW later on.

- **Near/Far.** You may not notice this at first but this movement changes the spectral characteristics of the sound. Normally as you move further away, the high frequencies drop off.
• Up/Down. Not easy to make happen in the studio but almost always suggested by the sound recorded. You will find that pitch inflection (glissandi) tends to suggest motion upwards / downwards and that the spectral characteristics of a sound will suggest height. Light whispers - indeed let us use the word ‘whispers’ as an audio example - suggest fleeting motion and a sense of height. Birdsong too suggests a certain lightness. Drums and other ‘heavier’ sounds are normally rooted to the floor. This ‘small is light will fly’ vs. ‘large is heavy will lie’ is not a hard and fast rule. Exceptions could include the sound of a helicopter for example. We know it is in the sky but its sound is often quite full on. Trevor Wishart cites the sound of the rainforest including whalesong. Clearly a spatial contradiction but an easy possibility in the world of sound.

1.2.2 What sounds can I record?

As mentioned earlier, all sounds are open to you to record but some will simply never yield good results in the studio. Here is a list of source files that composers at The University of Sheffield Sound Studios (USSS) have used in the past.

• Coins, Cowbells, Crotales (sustains and chimes)
• Hung Metals, Percussion, Pots and Pans (harmonic and inharmonic)
• Parcel tape, Velcro, Styrofoam, (clicks and rhythms)
• Stones, other assorted objects in cardboard boxes, paper crunch and tear (gesture texture)
• Cars, Street scenes, Rain and Thunder (environments)
• Human action, Human voice, Human situations
• Water pouring, Objects in Water, Streams and Rivers, (inside outside)
• Wine bottles, Wood Blocks, Western classical instruments (pitch noise)

1.2.3 Where does composition begin again?

Composition often begins with an idea. That idea is not normally a concept but a sound or collection of sounds. It is better not to drive your compositions from conceptual ideas or stories as making sounds fit into these forms is often extremely difficult. Better to happen upon a sound, accumulate a collection of similar or contrasting sounds and see if an idea springs from your close listening to these sounds.

In the past composers working at The University of Sheffield Sound Studios have used a number of western classical instruments as sources (Violin, French Horn, Piano, Zither). These instruments are already familiar to us from radio and concert settings. Taken in isolation they can also create a full spectrum of colour and dynamic.
1.2. COMPOSITION AND RECORDING

However you may wish for an even more restricted range of source sounds. Alistair MacDonald’s elegant work *Equivalence* (2007) is made up from the sound of bricks being gently rubbed together. Jonty Harrison’s *Klang* (1982) uses the sound of a casserole (lid and body). Adrian Moore’s work *Study in Ink* (1997) uses almost exclusively the sound of a whiteboard marker on a board (ouch!).

And the process is then quite simple. (Simple to state; difficult to implement; and often quite time-consuming to complete.)

![Diagram](Figure 1.2: The work-reflect process)

1.2.4 Reflect

Imagine the situation. The sound on your hi-fi is too loud. You need to make a qualitative adjustment, reach for the volume control and make the necessary change. You recognized a problem and reacted against it using tools specifically designed for the purpose. This is how the creative aspects of this book will develop. Taken to its extreme, you want to compose so you need to make sound; without sound you will have nothing. By contrast, if your piece/section is coming to an end you need to stop sound. Either way we are always reacting against what we perceive in any particular situation.

Although this book is not probing the physics of the ear and the psychoacoustics of listening, it is important to briefly mention how sound affects us physically and psychologically. Most fit and able adults have a hearing range from 20hz to 20,000hz (1 hz is one cycle per second). Sub-audio is more than often felt and seen. Supra audio is there but we just can’t perceive it. Other animals can, and do however. Our hearing is ‘tuned’ to perceive the frequencies most associated with our speech patterns. It will become immediately obvious once you start working with sounds that have strong frequencies in different parts of the spectrum that you will need to amplify some frequencies more than others. The most obvious will be the need to amplify low frequency sounds more than high frequency sounds (partly because you have to move more air for a low frequency sound - hence the reason why bass bins are so big and heavy!)

It is important to differentiate between intensity/amplitude and loudness, and between frequency and pitch. Intensity/amplitude is a measurable fact; loudness is our perception of it. Similarly, frequency is a definite; pitch is a rough approximation. When frequencies are tied together in harmonic (and
sometimes inharmonic) ratios, we tend to perceive the sound as pitched. It is the structure and intensity of these frequencies, (often called partials) that define the timbre of instruments. Our recognition of a source is often determined very quickly and is often related to the attack portion of the sound (where most energy is exerted). However, the sustained portion of a sound is also important in determining the size and shape of an instrument.

Many instruments - natural and synthetic - follow an ADSR shape (Attack, Decay, Sustain, Release), see figure 1.3. Moreover, when you come to make sound objects (small sounds with strong identity, often composites of a number of related sounds), you may find these objects being influenced by this profile.

![Figure 1.3: ADSR Envelope](image)

**Figure 1.3: ADSR Envelope**

Attack: time taken to rise to peak energy. Decay: time taken to decay to a steady state. Sustain: time remaining at steady state before Release: time taken to decay to the closure of the sound.

Farnell (2010) provides a quick yet comprehensive guide to the science behind our perception of sound.

And once sound has passed our inner ear and been converted to electrical signals, the brain takes over in ways that we still do not fully understand. We naturally process sounds quickly for similarity, difference, warning, survival, attack, defence, surprise, location, identification and discrimination.

We rely upon an innate understanding of (or at least a gut reaction to) the physics of the world to help us understand and break down sound as it happens. And in our musical world we are quite content to hear a phrase on the flute played by a human; content too (almost) with a never ending, humanly impossible passage played by a Sibelius-created MIDI file. However, ambiguous sounds such as those created by noise-making machines and electroacoustic synthesis patches may be so unfamiliar and sound so unnatural that we have enormous difficulty accepting them into our repertoire of living sound. On the one hand, as listeners we should remind ourselves that the physics of the world should not need to work in our imagination; and as composers we should consider that listeners may well expect to be led by the hand towards the complete unknown, lest they dismiss your work before it has even begun!
1.2. COMPOSITION AND RECORDING

1.2.5 Listening

As the author writes this, he is listening to a beat based music that simply keeps his mindset focused upon the job at hand. In fact, he is not really listening at all; this music is simply blocking out the knocks on the door, birdsong outside, the telephone ringing downstairs. In fact, many people use i-devices for similar reasons. (Though it is often baffling to see people with ear phones in while driving). Clearly we can direct our listening, focus our memory and alter our attention consciously as we search through sound in time (especially when listening to music). Writers including Schaeffer (1977), Chion (1990), and Farnell (2010) amongst many others, have come up with a concoction of terms for different listening ‘states’.

Schaeffer's (1977) modes of listening are as follows:

- Écouter: An analytical listening mode where we are detecting the event behind the sound.
- Ouïr: A passive listening mode where we just receive sound unwillingly.
- Entendre: An active listening mode where we tune into certain aspects or qualities of a sound.
- Comprendre: A more global mode where we perhaps begin to understand some form of communication in a musical language.

Farnell (2010) adds to these the following:

- Reflexive. Startle response, defence reflex to loud sounds.
- Connotative. Often associated with a fight/flight response. Base understanding without the listener having to think.
- Causal. Listening with an ear towards the physicality of materials and events (the rise in pitch as you fill a container with water being linked with how much empty space is left).
- Empathetic. Listening that is focused towards the emotional state of a being.
- Functional. Listening to understand the function of the sound (sonar and other sonification methods are functional)
- Semantic. Listening for the meaning behind the sound (a telephone bell for example).
- Signal Listening. Being prepared for a forthcoming sound.

Additionally, Farnell mentions reduced listening, analytic listening and engaged listening. These are the methods by which we begin to appreciate the music behind the sound constructions in sonic art. Reduced listening is the acousmatic approach; listening not to source or cause but to the sound itself,
its energy and flow. One can not do this without being analytic and engaged. Engagement is up to the listener (and as a composer it is well worth thinking about just how you help your listener remain engaged). Engagement helps listeners become more, or less analytic in their listening. It is important to remember however that you are a listener too. There are going to be times when you decide that you can dial down your analytical listening and just absorb the sound. Then there will be times when you need to become more engaged in order to remain interested. How you feel at any point in time, how much work you have to give to listening and how much reward you feel you are getting from listening equals VALUE.

There is of course a huge difference between the value of a listening experience and value judgements made upon your own sounds as you work with them. When you listen to sounds you tend to relate quantity (how much) to quality (how good/bad). Like any training, as you listen to more sound, make more sound and think about sound/sound shapes, your hearing will become attuned to the acousmatic-potential of a sound. Throughout this text you will be learning a number of creative processes to develop sounds (Chapter 2). These processes have a habit of producing quite specific qualities in sounds, qualities which, once recognised, need to be controlled. This is why the working method in this book is reactionary. For sure, there will be times when you perceive a quality and say “I need more of this” but with fundamental structures such as spectral shape, duration, texture and gesture, you will often find that the quality that is most apparent is the one that needs ‘taming’. As you develop sounds away from their original characteristics you may well find yourself working in loops. For example: as your long, continuous sound becomes increasingly shorter and discrete you find that you are now working with short and discrete and therefore require methods to create long and continuous.

Trust us, you will not end up going back to your original sound! Instead you will be in a place that is similar but different. Working out these similarities and differences and, how you got there and with what processes is composition. This book will suggest a multitude of pathways and recipes for a sound’s development and link composition theory, analysis and practice with technological skill acquisition.

If you are recording sound, you will want to spend some time tidying up your session and editing down your source files. There are no hard and fast guidelines as to how to edit down a session. However, we would suggest that you create files that have single sounds in them, files with similar sounds, files with gestures and files with textures. Naming files according to the source used (water1.wav, water2.wav) does not really describe what you have. Consider using onomatopoeia from the outset (waterbubble.wav watertrickle.wav) and remember it is wise to keep file names relatively short to begin with and make sure there are no uppercase, spaces and odd characters like %$^&*!. To extend filenames use only hyphens and underscores (watertrickle_reverb.wav or waterbubble-filter.wav). It is worth having a number of files that are very short (good for testing, but these might also be the basic elements of your material) and a number of files that contain all
manner of sounds (as we will introduce a number of techniques to ‘shuffle’ these files later on).

1.2.6 Analysis

Whether composing or listening, learning new tools or reading books like this one, the assimilation of knowledge so that you can draw upon it later is the basis of analysis. You will find throughout this book numerous representations of electroacoustic music: the waveform; the sonogram; the code snippet and the drawn graphic interpretation. All are clues to a greater understanding of what we/you hear. The majority are translations involving huge data loss (therefore presenting what is best presented according to the graphics available). Referring to the potential of the drawn graphic to render meaningful attributes, John Young writes:

> From the electroacoustic composer’s perspective, two fundamental questions are frequently asked during the creative process: what do I hear in these sounds, and how can the shaping and presentation of the materials I create convey something of my explorations on to the malleability of sound? If a completed work is a reduction and embodiment of the composer’s listening and manufacturing methods, analysis of that work is similarly not just located in the graphics or symbols that might be used to represent analogies, proportions or to sketch trajectories of development, but in their ability to embody something of the musical and compositional space that is articulated. Without that, the dark at the end of the metaphorical tunnel will remain for as long as we wish to continue. (Young, 2004, 8)

The vast majority of this book is about translation: translation from code to sound, translation from sound to words and translation from sound to image. There is every reason to assume that you will see shapes when making your music and indeed the visual analogies suggested later are all analogous to what you might expect to hear. Analysing one’s own work (especially while in the process of making it) is rare, though it might be an interesting process: composers need to keep more notes on intermediate steps if they think the preservation of their work is important (and especially if they want to repeat a successful procedure or recompose a work later in life). The detailed analysis of the works of others is part of the contextualisation of your own work and is of course vital.

1.3 Descriptors

We have mentioned before that Denis Smalley’s *Spectromorphology* has aided many composers when analysing sounds. Some have indeed considered how the reverse may be true, that Smalley’s terminology be used as a generative syntax (Blackburn, 2011). Manuella Blackburn’s vocabulary and diagrams are indeed most helpful in considering shape, trajectory and density but they are
for the most part procedural and her elegant diagrams for starts, middles and ends suggest all manner of variety, a variety which is not described in words. We need a multidimensional space to describe our sounds that ties in specific qualities and ‘gut feelings’. Stemming from two simple words defining a very expressive continuum, we start to home in on a number of descriptors based on metaphor.

- texture (metaphor to our sense of touch)
- gesture (metaphor to the body)

1.3.1 Metaphor

So we come to metaphor itself. Professor Bernard Hibbits (1994) gives us a very detailed description of how visual metaphor is used in American legal practice and how it is assumed to be stronger than any aural metaphor. He does however cite many instances where aural metaphor is used such as “speaking out, really listening, let the material tell you what to do”.

And here we have our abstracted syntax drawn from Simon Emmerson’s (Emmerson, 1986) grid.

![Figure 1.4: The relation of language to materials](image)

This simple graph describes how we work with material and what we try to say with it. In most cases when working with recorded sound, we are going to let the material tell us what to do with it (the syntax or method of composition, is abstracted or taken from the sound itself). And the nice thing about the sounds we are using (whether recorded in the studio or sampled from nature) compounded by the difficulties of being truly acousmatic, means we can shift quite easily between an aural or mimetic discourse. However, your position on the aural-mimetic discourse axis may well be different from others.

The really interesting thing about metaphor (and Scruton (1997) gives us a solid grounding here if you want to tackle some fairly heavy philosophical thinking) is that it provides a creative link with the composer/listener and tells us as much about the subject (us) as about the object (the sound). Two
music psychologists, Roger Watt and Roisin Ash (1998) considered how our
description of musical excerpts might give away clues about the listener as well
as the music. They presented their ideas of disclosure meaning in music. Part of
their experimental process was to force listeners into either/or categorisations
of music. Their thesis was that music can lead to similar reactions to those
we experience when encountering another person: that music can have specific
traits and states. This research had a profound effect upon the thought and
music of Adrian Moore.

We are now going to present a number of categories that you might find
useful in describing sound. Note now that they are all A/B pairs. It will often
be the case that you are likely to have something that not only exists between
A and B but that actively moves from A to B! We need to start simply so let
us think more theoretically to begin. Not all of the following descriptors will be
useful (some never!) but they are there as a guide. The idea behind this book
is that as you use some of this terminology to describe sounds and techniques
you begin to associate sound characteristic with useful techniques that may help
develop that characteristic.

<table>
<thead>
<tr>
<th>sex</th>
<th>season</th>
<th>reality</th>
<th>size</th>
<th>colour</th>
</tr>
</thead>
<tbody>
<tr>
<td>adult</td>
<td>summer</td>
<td>artificial</td>
<td>big</td>
<td>black</td>
</tr>
<tr>
<td>child</td>
<td>autumn</td>
<td>natural</td>
<td>little</td>
<td>white</td>
</tr>
<tr>
<td>boy</td>
<td>winter</td>
<td>real</td>
<td>broad</td>
<td>dark</td>
</tr>
<tr>
<td>girl</td>
<td>spring</td>
<td>unreal</td>
<td>narrow</td>
<td>bright</td>
</tr>
<tr>
<td>male</td>
<td></td>
<td>life</td>
<td>thick</td>
<td>dawn</td>
</tr>
<tr>
<td>female</td>
<td></td>
<td>death</td>
<td>thin</td>
<td>dusk</td>
</tr>
<tr>
<td>gentleman</td>
<td></td>
<td></td>
<td>heavy</td>
<td>day</td>
</tr>
<tr>
<td>lady</td>
<td></td>
<td></td>
<td>light</td>
<td>night</td>
</tr>
<tr>
<td>woman</td>
<td></td>
<td></td>
<td>large</td>
<td>evening</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sex</th>
<th>season</th>
<th>reality</th>
<th>size</th>
<th>colour</th>
</tr>
</thead>
<tbody>
<tr>
<td>adult</td>
<td>summer</td>
<td>abundance</td>
<td>huge</td>
<td>moon</td>
</tr>
<tr>
<td>child</td>
<td>autumn</td>
<td>lack</td>
<td>full</td>
<td>urban</td>
</tr>
<tr>
<td>boy</td>
<td>winter</td>
<td>empty</td>
<td>tiny</td>
<td>rural</td>
</tr>
<tr>
<td>girl</td>
<td>spring</td>
<td>full</td>
<td></td>
<td>red</td>
</tr>
<tr>
<td>male</td>
<td></td>
<td>empty</td>
<td></td>
<td>orange</td>
</tr>
<tr>
<td>female</td>
<td></td>
<td>full</td>
<td></td>
<td>yellow</td>
</tr>
<tr>
<td>gentleman</td>
<td></td>
<td>empty</td>
<td></td>
<td>green</td>
</tr>
<tr>
<td>lady</td>
<td></td>
<td>full</td>
<td></td>
<td>blue</td>
</tr>
<tr>
<td>woman</td>
<td></td>
<td>empty</td>
<td></td>
<td>indigo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sex</th>
<th>season</th>
<th>reality</th>
<th>size</th>
<th>colour</th>
</tr>
</thead>
<tbody>
<tr>
<td>adult</td>
<td>summer</td>
<td>abundance</td>
<td>huge</td>
<td>violet</td>
</tr>
<tr>
<td>child</td>
<td>autumn</td>
<td>lack</td>
<td>full</td>
<td></td>
</tr>
<tr>
<td>boy</td>
<td>winter</td>
<td>empty</td>
<td>tiny</td>
<td></td>
</tr>
<tr>
<td>girl</td>
<td>spring</td>
<td>full</td>
<td></td>
<td></td>
</tr>
<tr>
<td>male</td>
<td></td>
<td>empty</td>
<td></td>
<td></td>
</tr>
<tr>
<td>female</td>
<td></td>
<td>full</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gentleman</td>
<td></td>
<td>empty</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lady</td>
<td></td>
<td>full</td>
<td></td>
<td></td>
</tr>
<tr>
<td>woman</td>
<td></td>
<td>empty</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1.1: table of opposites
### Table of polar opposites continued.

<table>
<thead>
<tr>
<th>appearance</th>
<th>character</th>
<th>scene (position)</th>
<th>energy</th>
<th>growth</th>
</tr>
</thead>
<tbody>
<tr>
<td>clean</td>
<td>angel</td>
<td>background</td>
<td>unstable</td>
<td>accumulate</td>
</tr>
<tr>
<td>dirty</td>
<td>devil</td>
<td>foreground</td>
<td>stable</td>
<td>dissipate</td>
</tr>
<tr>
<td>clear</td>
<td>good</td>
<td>beginning</td>
<td>cold</td>
<td>changeable</td>
</tr>
<tr>
<td>cloudy</td>
<td>bad</td>
<td>end</td>
<td>hot</td>
<td>constant</td>
</tr>
<tr>
<td>complicated</td>
<td>brave</td>
<td>bottom</td>
<td>warm</td>
<td>closed</td>
</tr>
<tr>
<td>simple</td>
<td>cowardly</td>
<td>top</td>
<td>cool</td>
<td>open</td>
</tr>
<tr>
<td>dry</td>
<td>enemy</td>
<td>floor</td>
<td>excited</td>
<td>departing</td>
</tr>
<tr>
<td>wet</td>
<td>friend</td>
<td>ceiling</td>
<td>calm</td>
<td>arriving</td>
</tr>
<tr>
<td>hard</td>
<td>exciting</td>
<td>front</td>
<td>extreme</td>
<td>ascending</td>
</tr>
<tr>
<td>soft</td>
<td>boring</td>
<td>rear</td>
<td>moderate</td>
<td>descending</td>
</tr>
<tr>
<td>healthy</td>
<td>sweet</td>
<td>high</td>
<td>fast</td>
<td>forward</td>
</tr>
<tr>
<td>ill</td>
<td>bitter</td>
<td>low</td>
<td>slow</td>
<td>backward</td>
</tr>
<tr>
<td>liquid</td>
<td>fear</td>
<td>deep</td>
<td>harmonic</td>
<td>freeze</td>
</tr>
<tr>
<td>solid</td>
<td>courage</td>
<td>shallow</td>
<td>inharmonic</td>
<td>melt</td>
</tr>
<tr>
<td>gas</td>
<td>funny</td>
<td>near</td>
<td>focused</td>
<td>emerge</td>
</tr>
<tr>
<td>loud</td>
<td>serious</td>
<td>far</td>
<td>unfocused</td>
<td>disappear</td>
</tr>
<tr>
<td>quiet</td>
<td>gentle</td>
<td>horizontal</td>
<td>pitched</td>
<td>attack</td>
</tr>
<tr>
<td>mess</td>
<td>violent</td>
<td>vertical</td>
<td>noisy</td>
<td>release</td>
</tr>
<tr>
<td>order</td>
<td>happy</td>
<td>left</td>
<td>gesture</td>
<td>upbeat</td>
</tr>
<tr>
<td>new</td>
<td>sad</td>
<td>right</td>
<td>texture</td>
<td>downbeat</td>
</tr>
<tr>
<td>young</td>
<td>active</td>
<td>rising</td>
<td>continuous</td>
<td></td>
</tr>
<tr>
<td>old</td>
<td>lazy</td>
<td>sinking</td>
<td>discrete</td>
<td></td>
</tr>
<tr>
<td>pretty</td>
<td>noisy</td>
<td>up</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ugly</td>
<td>silent</td>
<td>down</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rough</td>
<td>polite</td>
<td>together</td>
<td></td>
<td></td>
</tr>
<tr>
<td>smooth</td>
<td>rude</td>
<td>apart</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sweet</td>
<td>powerful</td>
<td>under</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sour</td>
<td>weak</td>
<td>over</td>
<td></td>
<td></td>
</tr>
<tr>
<td>soft</td>
<td>public</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hard</td>
<td>private</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fat</td>
<td>safe</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thin</td>
<td>dangerous</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gritty</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>smooth</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pointed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rounded</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1.2: table of opposites continued

#### 1.3.2 Descriptors in real world situations

Clearly, no particular sound is going to say to you ‘I’m a lady, summer, hot and urban’. Some of these descriptors are going to be far more use than others, in
1.3. DESCRIPTORS

fact some you may never use! And this is by no means an exhaustive list. However, by linking a number of these descriptors to your sounds, and to processes, you might be able to marry acousmatic-potential and manipulation effect more quickly.

Take the sound of wine bottles for example. Banging them together produces an attack with very little delay, generating short sounds. Rubbing the bottoms of the two bottles gives a rough continuous sound. Letting the bottles bounce against each other gives an attack-release profile. The sound (where not an attack) is predominantly pitched but inharmonic. Because the material is glass, the sound is likely to be felt as harsh and bright.

So looking down this list, here are some explanations of how these terms might be applied to sounds.

Sex

You will find relatively little use for these descriptors but you may wish to think about Adult/Child when thinking about developing form. Imagine phrases and sub phrases that inherit from their parents. What characteristics do they inherit and how is the Child version different from the Adult version?

Season

Vivaldi’s ‘Four Seasons’ spring to mind but again you might wish to use these descriptors to hint at formal design with spring suggesting growth, winter suggesting death. To some these descriptors might arise from an over-riding suggestion from a piece. They are less likely to be attributed to individual sounds.

Reality

Artificial/Natural could be used to highlight the differences between a mechanically designed sound to more free-form material. Linear directions and static sounds often sound artificial. Sounds that move or change exponentially or logarithmically tend to sound more natural. You will find that volume curves in DAWs that are like the natural shapes shown in figure 1.5 tend to sound more realistic than straight lines.

Real/Unreal. You identify a real sound. A subtle transformation still keeping some semblance of the original may lead to this sound becoming unreal. Remember Trevor Wishart’s mix of whale song against the backdrop of a rain forest? Jonty Harrison includes the term surreal and perhaps the above example is as surreal as it is unreal. If one considers Salvador Dali’s melting telephones or human cellos, there is every possibility to create what is seemingly a real situation out of unreal objects. This real/unreal/surreal intersection has been used by many composers as a means of transporting listeners between worlds whilst providing ‘something to hold on to’ (Landy, 1994).
CHAPTER 1. WHAT IS SOUND?

Size

- Big/Little/Large/Small. These are very inaccurate metaphoric descriptors for volume or spectral power. One tends to describe small sounds as sounds with very little power, sounds that often lie in the high frequency portion of the spectrum. Small sounds tend to gather together (agglomerations) and fly apart (dissipate) without significant quantities of additional energy being provided.

- Broad/Narrow. Technically, these terms are normally applied to frequency bands (band pass filters might be broad or narrow) only allowing a certain range of frequencies to pass.

- Thick/Thin. This may be applied to the perceptive weight of the sound. If the spectrum is weighted towards the lower end, the sound may be thick. This may also mean that details are hard to come through (Imagine trying to draw fine details with a thick 6B pencil.) Equally, thin normally refers to perceptual content especially a texture that does not have enough power to motivate itself forwards.

- Heavy/Light. When applied to movement a heavy sound will be resistant to movement. Lighter sounds will probably want to be panned or spatialised in some way. Heavy/Light tie in with Large/Small but are not quite the same.


- Abundance/Lack. Terms used to focusing upon density. When you have heard or not heard something that you were expecting (within one sound or within a section containing a flock of different sounds) you might note an abundance or lack.

- Empty/Full. The perception of space or texture. This involves not only the definition of a space, but the location of occupants within the space. Moreover, it is the occupants which tend to define the space.
• • Huge/tiny. The perception of space, normally reverberation size. An exaggeration of Large/Small.

Colour

Synaesthesia is the condition where sounds trigger colours. Most people do not have a direct association between sound and colour. However, frequency spectrum placement has some attachment to colour. High equates to bright, low to dark. Blue is often associated with cold and red with hot. Blue is also associated with water, green with pastoral scenes. Day/night alludes to a sense of space and our perception of objects within that space. Urban/rural can be associated with machines/nature/darkness or green-brown-yellow hues. The mind always wants to use the eye, therefore it will often be hard for you not to conjure images, landscapes or worlds, whether they be real, unreal or surreal. If your landscape is sparse and large, chances are it is dark therefore individual colours will not make themselves known. A brighter world will highlight many more specific colours. To a certain degree, noise is known by colour depending upon the amplitudes of component frequencies. We call all frequencies at equal amplitudes white noise. As high frequencies decrease in amplitude we find Pink and Brown noise. As low frequencies decrease in amplitude we find Blue and Violet noise. Finally, if frequencies drop in the middle we hear grey noise.

Appearance

• Clean/Dirty. Often used to describe signal to noise ratio. A clean sound may well have been synthesized and be completely without background interference. Dirty sounds (since the rise of Glitch based sonic art) are rough around the edges and often appear to be untameable. They signify a cultural outcast, something that breaks the rules or does not conform to norms.

• Complicated/Simple. Normally associated with a perceived measure of textural density or sound object activity.

• Dry/Wet. Normally applied to technical issues regarding the ratio of original sound to manipulated output and most often seen in reverberation processes. A wet space affords multiple delays and reflections. A dry space leaves you only with your input sound and very few reflections.

• Hard/Soft. Normally associated with the perception of high frequency content in particular on attack portions of a sound.

• Loud/Quiet. The perception of amplitude.

• Rough/Smooth. There are potentially many areas where these terms may be used. Rough/smooth may be taken to differentiate the sound of stones versus the sound of a bowed glass (used as a metaphor for texture). They may be related to spectral variation. Equally they may be applied to
something as similar as a violin note played by a beginner without vibrato and one played by a profession perhaps with vibrato. The vibrato in this instance would be applied evenly so contributing to the perception of a smooth tone.

- Gritty/Smooth. Again used to talk about textures. Note that both words are highly onomatopoeic.

- Pointed/Rounded. Can be applied to both attacks (like soft/hard) or phrases, sections or statements. Again, these are highly onomatopoeic words.

Character

This section is all about higher level functions within your sonic art especially the emotional response that a sound or sounds may trigger. They are all very ‘human’ characteristics and might suggest very formal approaches to listening. Perhaps most obvious for use are the following:

- Exciting/Boring. Possibly pertaining to perceived speed of a section

- Funny/Serious. Sound choice and juxtaposition can often generate humour.

- Gentle/Violent/Powerful/Weak. We use these terms to describe the power of the wind. That degree of turbulence in audio is entirely possible.

- Happy/Sad. More obvious in tonal music where happy equates with major mode and sad with minor mode music. However, in electroacoustic music pitch inflection can often indicate this state with a downward slide tending towards emotions of sadness.

- Public/Private. This may well correspond to a near/far spatial position indicating a degree of intimacy. Filtered voices often tend to sound closer and therefore more personal and private to the listener. Small sounds may well be private. These terms therefore tend towards descriptors of form rather than content.
• Safe/Dangerous. Composers may well have experienced feedback gradually moving out of control. It is possible to formulate a nervous textural or gestural energy that is suspect and dangerous. In glitch based music, the drone that sounds like a jack plug halfway out is also particularly ‘dangerous’.

Scene

This section has far more literal consequences for the description of materials.

• Background/Foreground. Concerning sound in space, this clearly is important for changes in perspective possibly involving reverberation or high frequency content. This could well be a structural definition where both background and foreground exist at the same time and go to define the space.

• Near/Far. Slightly more poetic and subjective view of background/foreground perhaps more to do with motion.

• Beginning/End. Pertaining to shape not just in terms of a piece but phrase and sound-object. Beginnings may well be full-on downbeats or have an upbeat or anacrusis immediately prior.

• Bottom/Top/Floor/Ceiling. Normally referring to position in the frequency spectrum. Using these terms it is possible to consider canopied or rooted settings (pedal points etc.) where low/high drones support material within the space.

• Front/Rear. Normally used in performance where sound is projected in a 3D space.

• High/Low. Frequency content or pitch. Perhaps the most obvious ‘positional’ element.

• Deep/Shallow. Rarely used probably because of their analogy with water.

• Horizontal/Vertical. Possible descriptor of shapes in time relating to frequency more than any other parameter.

• Left/Right. Panning and spatialisation.

• Rising/Sinking. Descriptors for pitch glissandi.

• Up/Down. More aggressive descriptor for frequency spectrum movement.

• Together/Apart. Structural placement in time. Formal perception of structural divergence/convergence.

• Under/Over. Relationship of one set of material to another (often associated with canopied/rooted settings where A is rooted under B for example).
CHAPTER 1. WHAT IS SOUND?

Energy

This section is again more subjective but is based upon a number of common features of sound in time and space.

- Unstable/Stable/Excited/Calm. Normally the perception of textural content but equally can be applied to a sequence of gestures.
- Cold/Hot. Perception of frequency content and subjective assessment of landscape.
- Fast/Slow. Perception of speed in terms of texture or gesture.
- Harmonic/Inharmonic/Focused/Unfocused. Perception of pitch whether singular or streamed.
- Gesture/Texture. Perception of structure in time
- Discrete/Continuous. Perception of time often related to our ability to ‘chunk’ data.

Growth

Perhaps the most important (and therefore potentially incomplete) section, growth leads us to think about structure.

- Accumulate/Dissipate. The addition/subtraction of materials and energy.
- Departing/Arriving/Open/Closed. Structural devices that are the result of material leading up to or moving away from either a beginning or an end.
- Ascending/Descending. A slightly more objective view of pitch glissandi and better terminology than up/down.
- Forward/Backward. Pertaining to the basic perception of time (Remember that reverse envelopes are rarely used except for situations such as pre-reverberated sounds.) Possibly also as a structural descriptor where sections may move forward and backward in time (ABCBA)
- Emerge/Disappear. Perception of amplitude but more subtle than start/stop.
- Attack/Release. Perception of energy profile, normally in sound objects

1.4 The Sound Object

The sound object is commonly understood to be a composite object made up of a number of smaller sounds, often with a global ADSR shape and as Manuella Blackburn (2011) points out, emergence, transition, release envelopes or upbeat,
statement, disappearance envelopes. However, the sound object is considerably more complicated than these two or three archetypes.

At a basic level we can define sounds emerging from the distance and moving towards a closure

![Figure 1.7: Emerge release profile](image)

Or sounds with more potency, marking their space more dramatically.

![Figure 1.8: Upbeat profile](image)

Gradually we realise the composite can be in both the X and Y directions.

![Figure 1.9: Downbeat profile](image)

Leading towards a complex object.
CHAPTER 1. WHAT IS SOUND?

Additionally, each block of sound may well be an object in itself. If the sound is not a composite then it may simply be shaped by an amplitude envelope or a change in one particular spectral parameter.

*Top tip.* Always remember a similar but different policy to working with sounds and remind yourself that unless you are working very mechanically, nothing ever stays still (even silence has a direction).

For example take the following drone based sound:

Example 1 in figure 1.11 is rather boring and mechanical with no loss or injection of energy. Potentially usable but quite difficult to lead into and get out of. We might consider adding some undulation to the drone. Example 2 has turned into something completely different and the up-down roller-coaster ride we are taking has the same degree of redundancy (potential insignificance) as the static drone. Example 3 however, stays with the overall shape of the original drone but adds in a couple of small gestural elements and gently sways the drone with musical expansion and contraction. Imagine curtains blowing in the wind from an open window.

Sound objects can then become the thematic motivators for your sonic art. Their onsets are powerful enough to trigger continuant sounds, from which you can conclude a sound object or move into something more textural.

Consider the following example:
Here the opening object supplies sufficient energy for a steady state drone or texture to emerge and for a rising glissandi to reach a steady state in parallel to the original drone. From this comes another sound object that saps the energy of the drone, killing it off, leaving each drone portion to fade naturally into the distance.

Sound objects need not always be of high intensity or contain large dynamic range. One particular trick to help foster the generation of a multitude of small sound objects is to take the undulating motion of one soundfile (possibly a texture or drone) and use the points of articulation as a trigger for a particular sound object (a continuation object). Denis Smalley calls this kind of material ‘texture carried’ because the texture literally carries the gestures that lie thematically above it. But because these gestures appear (or should appear) to spring from the texture they almost become de facto sound objects.

In this example the continuous waveform has a number of large and small perturbations which could supply sufficient energy to allow for the addition of other material that maps with the Attack, Decay, Sustain, Release profile of
the texture. The idea here is that at each point we feel that the drone is being articulated by the gesture/sound objects and not the other way around! Where this fails and you hear a distinct separation between the texture and the objects, the working method is no longer continuous but discrete with the texture being A, and the individual objects, B,C,D,E, etc. If this does not feel natural then this will not be a good experience for the listener.

In a similar way, gestures may frame particular moments in time. As gestures get smaller and/or become more dense, the boundary between clear identification of something gesture-framed or texture-set may become blurred. Denis Smalley surmises thus:

Both gesture-framing and texture-setting are cases of an equilibrium capable of being tipped in either direction by the ear. They indicate yet again areas of border tolerances, ambiguities open to double interpretation and perceptual cross-fadings depending on listening attitude. On the whole, gestural activity is more easily apprehended and remembered because of the compactness of its coherence. Textural appreciation requires a more active aural scanning and is therefore a more elusive aural art. (Smalley, 1986, 84)

1.5 Gesture

Our discussion of sound objects has naturally taken in the idea of gesture, as sound objects are notably so marked. Gesture is at the heart of the sound object. It is the perceived intention of your sound object or at a lower level, the action of a (normally) human agent to generate a sound by striking, hitting, rubbing, or by shouting, whistling or breathing. Denis Smalley talks about the degrees of removal from the human-ness of the gesture (First, second and remote surrogacy, (Smalley, 1996, 85)) but in the studio there are a number of key gestures that you should consider trying to create:

• Attack with quick release. This is often delivered by a synthetic patch that has a very tight envelope or by recording attack-based sounds: drum hits, plucked strings, vocal exclamations. In many circumstances you are going to get some sort of decay attached to your attack. Don’t chop this off; it adds to the natural feel of the gesture.

• Attack with multiple contiguous starts and ‘torn flag’ ending.

![Figure 1.14: Attack with ‘torn flag’ ending](image-url)
1.5. GESTURE

- Swipes. A swipe has a logarithmic shape that starts slowly and gets increasingly faster. A more onomatopoeic interpretation might be *swish*. A swipe or swish at the start of an object normally falls over into an attack. A swipe or swish at the end of an object normally acts as a full termination (perhaps with reverberation).

![Figure 1.15: A quick filter swish or swipe with frequency and amplitude shapes](image)

- Acceleration / Deceleration methods. Similar motion to the swipe/swish but with discrete components rather than continuous sounds.

![Figure 1.16: A chirp based on Farnell (2010) with acceleration and deceleration motions](image)

If gestures are considered as potential ‘motifs’ of our work, we may well create gestures with different durations of attack and resonance, different component parts, multiplying or adding to our original. This may help create structure in a larger phrase.

Because each gesture has a different energy and speed, we may create a ‘melody’ of gesture, stringing gestures together one after the other (or with slight overlap). We really do need to be in careful control of the ebb and flow of energy here else our phrase becomes nothing but a series of attack/resonances. It is increasingly rare in electroacoustic music today to hear bold gesture in
1.6 Texture

When talking about texture, please refer to the tables printed on pages 15 and 16 of this chapter. Textures tend to arise from mixing streams of material. This material has often been manipulated in a granular fashion in that portions of an original have been selected and repeated (with modification) many, many times. Granular synthesis allows you to select a small portion of sound and extract it like ‘audio-DNA’. In most granular synthesizers you can select a grain’s size, pitch, location from the original, rate of repetition and placement in the final mix.

![Granular synthesis illustration]

Figure 1.17: An illustration of granular synthesis to create a time stretch texture

![Randomised input pointer]

Figure 1.18: Randomised input pointer delivering hyper-mixed resultant

Remember that each time a grain selection is made the output grain may be changed in pitch. Depending upon the size of the grain the resulting texture might bear strong similarity with the original or it may sound as though the spectrum of the original has been captured. Large grain sizes of greater than 250 milliseconds tend towards a technique more commonly known as brassage (or micromontage) where the sounds taken are still recognizable. As the grain size decreases, we move towards more drone-like textures that are potentially smoother (depending upon the pitch variation of the output grain). In most programs that allow granular synthesis, parameters can be varied or randomized. Some settings will be better than others for the sounds you have. Vocal sounds

---

2 For a more comprehensive discussion granular synthesis please see Microsound by Curtis Roads (Roads, 2004)
are particularly good with granular synthesis. Remember also that if your input sound has silence in it, chances are your output sound might have silence too. This is a great way to create randomized ‘splatterings’ of sound.

As figure 1.19 suggests, granulation can create textures that are solid or more ‘bubbly’ in nature. The analogy with an airbrush works quite well. The brush can vary in density, vary in distribution, vary in volume and consequently create a texture that is (from top to bottom) dense pointillist, airy pointillist, string-of-pearls, continuous undulating, same but louder, continuous steady-state. As stated previously, at no point do any of these diagrams sit absolutely still; even the bottom line has a variation in density. Absolute repetition is just so difficult to get out of. If you allow for the possibility of change, you give yourself an exit strategy.

Granular synthesis is enormous fun as you can create huge quantities of sound from very small quantities of sources. The acoustic DNA extraction allows you to play your source and continue a drone-based texture alongside it with ease.
1.7 Landscape

When we listen to individual sounds whether in a reduced listening mode or not we are looking for some reason for a sound to be ‘there’. The best reason (musically speaking) is its relation to sounds that have gone before or that are coming after. Consequently we are well suited to creating a landscape within which we place our sonic ‘inhabitants’. As we walk out of the door in the morning, we immediately get a grip of the sonic landscape in terms of perspective (our next-door neighbour’s car versus the cat across the street, versus the milk-float down the road, set in the mid-20s heat (if in summer)). It is this final characteristic that is most interesting. The perception of heat and the feeling of an ‘open, almost relaxed’ environment sets the size of your environment (very few players, large space). Trevor Wishart (1996, 140) breaks down the perception of landscape into three components.

![Figure 1.21: Wishart’s three components of landscape perception](image)

Whilst we can use time and frequency cues to assess the shape and size of any real or created space, we are most interested in what that space then suggests to us from a programmatic or dare we say it, romantic point of view. This is where, for some, as composers and listeners, we can for example move quite quickly from a relatively objective description of size ‘large, spacious’ to a subjective appreciation of ‘barren’. Barren immediately suggests a colour; there’s not much light here and compositionally, potential techniques suggest themselves. Sounds can come out of the darkness, make themselves present and fade back into the mist. Like twinkling stars in space or mysterious figures moving around a misty graveyard in the middle of the night, our perception is hazy therefore our sounds need not be in focus at all. We, as composers can hint and suggest and save the real personal introduction for when the scene changes.

---

3Reduced listening is the technique that allows us to adopt the acousmatic stance, listening to sound without reference to source or cause. The repetition of a sound or passage often tends towards the acousmatic as we are led to shift our attention from surface or superficial details to the interior qualities of the sound.
1.8. ENVIRONMENTS

Consider too the bustling city landscape. Noise. Not just literally but metaphorically. All manner of colours, all manner of shapes and sizes rushing past us at speed, not just loud but multiple changes in volume. Now consider the photographer’s trick of zooming out so that sounds that were around you in three dimensions now appear as though on a two dimensional canvas.

Landscape comes out of texture and gesture but texture in particular plays a key role in defining the world in which we are listening. Wishart, Harrison and others talk about the real, unreal and surreal in their landscapes (or environments - perhaps heard as a more cohesive filled landscape), especially when playing with our ability to recognise and formulate a sense of place and our understanding of scale. Landscape as seen here is much more to do with a painter’s vision, not a narrative, and is much more to do with background than foreground. In many respects it is a few trees gently swaying in the wind placed on the brown forest floor, set against the rocky mountains, behind which the sun gently sets against an azure sky. Or it is a heat haze obscuring the simple oasis set against the rolling sand dunes of the desert. Or, consider the countless apocalyptic scenes from every TV movie you have ever seen. Skeletal structures, set against a backdrop of decay. As one final, comedic example, consider space. The vast emptiness as we contrast a near-focus earth with a distant moon. Or a moon-scape with spacecraft fly-by. Where is the camera, where are the lights? Deep in the vacuum of space (potentially at absolute zero not to mention with absolutely no sound whatsoever). It seems we can suspend disbelief quite easily. We should therefore question why we have difficulty using our visual imagination when presented solely with audio.

1.8 Environments

This is perhaps what Wishart refers to as landscape; the scene with background, foreground, major players, walk-ons, cameos, props, lights, camera, action! However, we should try not to consider the real world at all as, chances are, we have recorded some water, some drums, some wine glasses and we are working an a very abstract domain. Environments such as the above can be honed down to something very abstract too. Foreground, background, big, small, near, far, up, down, left, right, light, dark. An environment in electroacoustic music needs to be plausible. And in many respects this implies some concept of nature to exist. Perhaps more than this, it might be possible to draw upon the concepts of object-oriented programming to shed light upon our process of constructing sounds, sound objects, gestures, textures, landscapes and environments.

1.8.1 Object-oriented concepts

This is not a proposal for a composition system. This whole book is about fusing sounds, composers and tools together and seeing what happens. However, you have already seen from the description of sound objects, gestures and textures
that ‘anything goes’ is just not acceptable. In a plausible environment we may well have sound objects that are related (in any music you will expect to be able to ‘chunk’ data and ‘relate’ it to other ‘chunks’ during the course of a work and electroacoustic music is no exception). They will relate to each other, adapt or sit well in the landscape and have a sense of growth or decay. Energy will arrive or drain from the system. The sun will rise and then it will fall.

Object-oriented concepts include:

- **Object/Class.** A class may be trees; an object one individual tree. Methods are procedures that act on objects and may include growth patterns, number of branches etc. Functions are more to do with how objects work within the environment (in this instance, how the branches sway in the wind).

- **Inheritance.** Consider a class of cars. The class of trucks may inherit from cars and bring their own additional functions and methods.

- **Polymorphism.** The ability for an object comprising many sub-objects to react in different ways depending upon how they are called.

### 1.8.2 Naturality and plausibility

Although not hugely constructive, the natural focus of object oriented design is a helpful metaphor to consider plausible environments. Our related sound objects fit neatly within a class structure. Our textures or drones (created out of sonic DNA drawn from objects) are inheriting certain genes and turning into completely different (but related) creatures. Our sounds (as we will see in chapter 3) will respond differently to different processes (from ‘very well’ to ‘not at all’) depending upon their sonic content (as described in this chapter).

Landscape and Environment are particularly energizing when heard in full surround sound (or multi-channel works). However, it is just as useful to consider the stereo space as even here, it is possible to create a fully immersive environment. In addition to setting the scene, laying out a number of characters and injecting energy in to the system composition becomes an animation process, attributing behaviours to sounds so that they react in potentially fruitful and musical ways with each other and with their landscape.

Our discussions on Landscape (section 1.7) and Environments (section 1.8) will figure more in later discussions concerning structure (section 3.3, page 81) and form (section 3.4, page 83).

### 1.9 The Continuum

In previous examples we mentioned how gestures might gradually accumulate into a texture and textures might dissipate into gestures. When considering the list of poles used to describe sounds earlier, our sounds are rarely going to be at the extremes of the continuum. Instead they will be somewhere towards
one or the other pole or traversing the continuum in a multi-dimensional space. It is for us to find the acousmatic potential of a sound that suggests further transformation or take a brute-force method of applying a process to shift one sound out of its dimension of least resistance into another realm. An example here is taking a noisy texture and creating a very smooth pitched sound by using resonant filters. Such is the wrench away from noise that a subtle crossfade between original noise and pitched effect often appears very noticeable.

1.9.1 Morphology

Seamless morphologies in the continuum between chalk and cheese are practically impossible. A morphology between ‘ssss’ and ‘zzzz’ is however distinctly plausible. Unfortunately the wow factor compared to our well known on-screen morphs is nowhere near as potent. The continuum, like any attribute, is well worth exploring and manipulating to the point where the listener recognizes its use.

1.9.2 Composition

Composition therefore is magic: twisting sound to your design, leaving a plausible trail for the listener to follow. You need to show enough to draw the listener in but not too much that they know precisely what is coming next. The frustration of expectation is a very useful axiom in this instance. Again, frustrate the listener for too long and your labours will be in vain; give the game away too soon and the direction of your work is weakened. Composition itself rests uneasily within a multi-dimensional continuum.

1.9.3 Performance

Of course once your piece has been completed you may present it in performance and this may involve you thinking about Sound diffusion. Although this book is about composition, the final presentation of your work will influence the composition of the work from day one. Sound diffusion is well documented especially by composers such as Jonty Harrison (Harrison, 1998) who have spent considerable amounts of time and money creating loudspeaker orchestras that enable composers to diffuse (or project) their sound into a space, giving audiences an accurate and exciting listening experience by rendering imagined textures, landscapes and environments initially encapsulated within a two-dimensional stereo listening space into a real three-dimensional space with loudspeakers positioned all around the audience. You may find that some sounds that feel ‘high, fast and fleeting’ and which you perceive as being ‘in the sky’ when listening over headphones can actually be placed in loudspeakers well above the audience so giving them an extra sense of life. Part of chapter 3 reflects upon the relationship between a sound’s descriptors, its potential development, its potential place in a mix and consequently its place in performance (loud, soft, high, low, fast, slow, front, rear, static, moving and so forth).
Chapter 2

What does all this software do?

Student: What does all this software do?
Teacher: Ah, anything you want.

2.1 Introduction

2.1.1 The flip side of anything is nothing at all

Having some idea about what a sound may be saying to you, you might now want to come back at it and force some change. However if you are completely new to the software mentioned in this book, you might like to have a look at the appendix or take apart some of the tutorials within the Pd distribution. We would suggest trying Pd before experimenting with Blue/Csound and Supercollider. You have to start somewhere. We have actually made the manipulation of sound in the first instance quite easy by making a number of ready-made objects that perform specific functions. Once you become more familiar with the tools on offer, you can open up our patches and adapt them to your needs. You will actually find that many of our tools emulate the techniques and processes found in the help files as this, not surprisingly, is where we began too. We start with one process at a time, but remember not only can processes be joined together in series (one after the other) or in parallel (one alongside another), but a sound can be processed many times through multiple processes. But before you think you will end up processing yourself into oblivion, hopefully you will hear a finite end to a sound’s potential or that sound A just does not work with process X. Some processes are so ‘invasive’ that they identify themselves quite clearly. Be wary of this as it could lead to clichès (which is not necessarily a bad thing!)
CHAPTER 2. WHAT DOES ALL THIS SOFTWARE DO?

2.2 The USSS Pd-toolkit

2.2.1 Plug and play

With very little Pd experience you should be able to get sounds into and out of the usss toolkit. Open a new file (File: New) and save it immediately somewhere useful. Having downloaded the files that go with this book including the usss toolkit\(^1\) make sure that you point Pd in the direction of the ussstools folder. Normally this can be done as a relative directory path, ‘go from where I am, out of my folder and into the ussstools folder’ (as shown in figure 2.1). Make an object box (ctrl+1) and type the following:

\[
\text{declare -path ../ussstools}
\]

Figure 2.1: relative declare path

If however you want a direct path to your ussstools you might type:

\[
\text{declare -path /mydrive/myhome/mypdfolder/ussstools}
\]

Figure 2.2: direct declare path

You may need to quickly save and reload your new patch in Pd so that the path can be found. The folder ussstools when opened should contain all the Pd patches and helpfiles. An object box looking like the following should find the necessary ready-made patch. Note that usss.sfplay is the name of the object we have made: mysfplay is the name of this ‘instance’ of the object. You can have multiple players but they must have different names.

\[
\text{usss.sfplay mysfplay}
\]

Figure 2.3: calling a ussstools object box

Whilst you can work through the examples in the book, if you type usss into an object box, right-click and select help, you see all the objects we have made. Right-clicking and selecting help in each of these objects should bring up a working help file.

\[
\text{usss}
\]

Figure 2.4: a patch containing all the tools

\(^{1}\)http://www.shef.ac.uk/usss
2.2. Input, Output

The input module normally takes the sound from your microphone but you may need to see appendix E if you are working with Jack for example. Connecting interfaces and working with sound drivers is never quite as simple as plug and play. The sfplayback object requires you to load a sound (selectfile) and then check 1 for play, 0 for stop. Note too that the names mysfplay.play can be placed in send and receive boxes so that playback can be automated.
2.2.3 Envelopes

The envelope tool requires you to draw in a simple envelope. For this to start you need to place the mouse at the bottom left hand corner of the drawing area (as marked by a cross in figure 2.6). You also need to specify a duration for the envelope to run its course (total_time). Finally, make sure you trigger the envelope with a bang. More creative use is made of the envelope tool in figure 2.29. This is quite a powerful tool when the bang is automated by say, a metronome or the amplitude of another soundfile passing over a threshold.
2.2. Filters

usss.bandpass

Figure 2.7: usss.bandpass

Please use the helpfiles to figure out what each parameter does. The bandpass filter can really 'squeeze' a sound's profile. A telephone is like a hardware version of a bandpass filter. Because the phone’s loudspeakers are so small and inefficient, the spoken voice ends up sounding considerably weaker through the phone. It can be used therefore as a means of spectrally fading sounds out (by sweeping one side of the spectrum towards the other) or as a means of creating ‘waves’ of sound by sweeping the centre frequency with a constant bandwidth. Once you have a thinner, filtered version of a sound, you may well consider mixing this with other sounds creating strata.

usss.combfilt

The comb filter is a very basic tool that is normally used to colour a noise-based sound on or around a resonant frequency. Often you find comb filters stacked in parallel so that you can create chords. The well known GRMTools Comb filters allow you to stack 5 filters in parallel. You can vary the pitch (expressed as a MIDI note) and the intensity of the filter in this patch. If you want to make multiple instances, this is possible. A sequence of numbers can be unpacked to the farthest right inlet of each instance to create chords.
CHAPTER 2. WHAT DOES ALL THIS SOFTWARE DO?

Figure 2.8: usss.combfil making a C major chord

usssl.fftfilter

The fft filter is more like the bandpass filter than the combfilter in that you draw
the shape of the area under which you want the frequencies to **pass**. However,
in your drawing you are not only selecting frequencies (very broadly) but also
adjusting the amplitude of those frequencies. This is because the patch uses an
fft which splits the input sound into component frequencies and amplitudes.
As with the envelope function, you need to place the mouse on the baseline
(in the far left hand corner) so that the mouse icon turns 60 degrees clockwise
before drawing. However you can write algorithmically to the graph. Please see
the functions below which make interesting frequency selections.
2.2. THE USSS PD-TOOLKIT

Figure 2.10: random filters

Figure 2.11: spaced filters

Figure 2.12: spaced filters decreasing amplitudes
CHAPTER 2. WHAT DOES ALL THIS SOFTWARE DO?

usss.filtersweep

Figure 2.13: usss.filtersweep

Figure 2.13 with the values approximately as shown, puts a sawtooth wave at sub-audio through a swept rising filter. The phasor^~ becomes series of clicks with each click containing all frequencies so activating our filter effectively. The filter’s Q or quality is high so the filter only lets a very small frequency range pass. Please note that this patch is mono input and output.

usss.reson

The resonant filters here are not dissimilar in output to the comb filter. This patch is modelled very loosely on the GRMTools Reson plug-in.

Figure 2.14: usss.reson

There are 16 filters in total. You can send a harmonic series as a list (100
2.2. THE USSS PD-TOOLKIT

200 300 etc.) in the right-most inlet. Once you are familiar with Pd you may want to adapt this patch so that you can specify the number of filters (although this is quite difficult to do). However, this patch clearly places a 'sheen' of pitch colour on any sound with moderate to full spectrum activity.

Remember that the filters above are quite 'invasive' and are clearly different in style from traditional equalisation or eq. Included in the toolkit are standard high-pass and low-pass filters that you see in traditional DAWs. Eq is often used to subtly adjust particular frequency areas of your sound. It is a tool that can be used anywhere in the processing stage; alongside the editing process or towards the mixing stage. Alistair MacDonald recommends that all sounds should receive some Eq, even just to 'hear' more clearly the spectral content of the sound. (see appendix B.2 on page 90)

2.2.5 Granulation

usss.granular

Granulation is cited periodically during this text as being one of the easiest and most efficient ways of generating material. Granulation was always with us as a technique as it is essentially repetition but on a small scale, often with small or gradual change.

Figure 2.15: usss.granular

Figure 2.15 shows a granular patch. You can load in a sound directly into memory or record live. Position is the position in your soundfile from zero (start) to one (end). This you can randomise. The grainsize is the small ‘grain’ of sound shown in figure 1.17 and figure 1.18 that is repeated. This
CHAPTER 2. WHAT DOES ALL THIS SOFTWARE DO?

Grain can be altered in pitch and this pitch can be randomised. The grain gain is normally put to 1 as without this you will have no volume. Finally, the grain pan moves from 0 (mono) to 256 (grains placed randomly in the stereo field). This is normally the default setting as it is rare that you would you want a mono output. In this figure you also see a metronome that scrolls at your speed through the position. This emulates a very basic timestretch if you slow the speed right down (and this means increasing the number going to the metronome which ‘ticks’ every X milliseconds). If your input sound is 10 seconds (10,000 milliseconds), to create an at pitch playback you would need to create a ‘tick’ every 100 milliseconds. Granulation across multi-channels (more than two) allows you to explode your timbre in the space, a truly fascinating concept. You will note that most of our tools are for stereo soundfile creation. They can all be adapted quite easily to suit multi-channel output. Composers at Birmingham University associated with BEAST (Birmingham ElectroAcoustic Sound Theatre) have made a toolkit for multichannel sound file development in MaxMSP (BEASTTools).

2.2.6 Pitch and Frequency shifting

ussv.varispeed

Varispeed emulates the old fashioned ‘tape machine’ where as you speed the tape up, the sound gets higher in pitch and as you slow it down, the sound gets lower. However, as with all music, there is a very close link with mathematics. You know that the A natural the orchestra tunes to is 440Hz. The A, an octave above is 880Hz.

440hz 880hz

Figure 2.16: Musical notes and frequencies

Not surprisingly then, as your tape speed doubles, all frequencies increase by one octave, as it halves, the pitch drops by one octave. If you are not dealing with pitched sound, all you need worry about is the maths. Consider the sound in terms of frequencies (100hz, 700hz, 1500hz). As the speed increases to 1.2, you will expect to hear frequencies of 120hz, 840hz, and 1800hz.
2.2. THE USSS PD-TOOLKIT

Pitchshift will transpose your sound without changing the speed. This is done in the time-domain, so as your transposition reaches extreme levels, distortions will appear (however, they are often interesting and beguiling, though perhaps quite well used by composers). Remember to lift the gainL and gainR faders. The window function can be seen as a grain. As the window exceeds 100milliseconds, you hear explicit repetitions of sound. As it falls below 40milliseconds, the timbre of the input sound will change dramatically. However, for ‘quick and dirty’ transposition without changing speed this patch works extremely well. Once you are familiar with Pd, open this one up and try to figure out how it manages to shift the pitch using delay lines. The doppler shift taken to extremes!

2.2.7 Spatialisation, Delay based effects

Spatialisation is rolled up in your very first recording so it is hard to consider it as an independent effect. Placement of sound can be defined at grain, sound and sound object level and may be fixed or moving. At a structural level
(sound, sound object) placement and motion deliver momentum and create landscape and environment. Be wary of placing long mono (or spatially thin) sounds in very specific, non-central areas in the stereo space as, especially when wearing headphones, this effect can become tiresome. However, whenever you are working with streams of sound, panning is a very useful structuring device that works well with a variety of metaphorical descriptors. Short impulses of sound often suggest one particular ‘place’ in the stereo field. Their careful positioning will allow you to play more impulses faster and create a sense of structure and meaning to the space.

![Figure 2.19: Four panning profiles](image)

In figure 2.19 A we see a \textit{closed->open} metaphor as material (either sounds with a stereo image or individually panned particles) moves from constrained to free (or focused to dispersed). The opposite also works well and can be seen in figure 2.19 D with an ‘exit stage right’\textsuperscript{2} profile. Figure 2.19 B represents a meandering but focused image while figure 2.19 C suggests \textit{far->near} or an approach to importance. Your sounds, particularly in this last case (C) need to be able to define the space within which they move quite well. They will need to be identifiable and will probably have a well defined spectral shape (in short, be a little bit noisy, possibly comprising discrete, repeated sounds) so that you can pick out the movement. Try panning a 100hz sine wave compared to a 100hz pulse wave to realise just how much the ear needs some (high) frequency content to articulate spatial movement.

\textsuperscript{2}Remember that actors - and the same is true for sounds - never walk off ‘stage centre’. Also ‘stage right’ is from the point of view of the actor not the audience.
There is very little to say about reverberation here as it is quite a well understood phenomenon. Some reverbs ask you for a reverb time, others talk about damping. If there is very little damping, you are likely to have a larger reverb time. Roomsize, damping and width all effect the quality of your reverb. You can then adjust the balance of ‘real’ with ‘reverberated’ or ‘wet’ signal. Finally, as an added bonus, the freeze provides a very accurate acoustic snapshot (not dissimilar to sitting on one small grain in usss.granular).

Delays are at the heart of reverbs on a small level, and on an even smaller scale, filters. They are also used to create chorus, phasing and flanging effects. As
with other patches, the `level` needs lifting before you hear sound. Extreme effects can be heard as you move the `modfreq` and `moddepth` far from the left hand side of the slider. Beware also of feedback when using values greater than 0.5.

**uss.s.panner1**

This panning patch works on a mono signal. This patch is useful if you want to create specific patterns of sound movement. When working with a stereo signal you may wish to tilt and sway the balance of the image. This may be as simple as multiplying the left and right signals by different amounts.

![Figure 2.22: uss.s.panner](image)

In the example above (figure 2.22) both the left and the right signals of a stereo file are going to different panners. The sliders on the right allow you to set the width of the stereo and then pan that width, so enabling some of figure 2.19. This kind of accurate panning is often better achieved with automation in your DAW however.

### 2.2.8 Synthesis

![Figure 2.23: sine wave oscillator](image)

These and other waveform types are demonstrated in `uss.s.waves.pd` where a phasor going from 0 to 1 is transformed according to mathematical rules.
2.2. THE USSS PD-TOOLKIT

\[
\text{expr~ if ($v1 > 0, 1, -1)}
\]

Figure 2.25: square wave

\[
\text{expr~ if ($v1 < 0.5, v1, 1-v1)}
\]

Figure 2.26: triangle wave

Figure 2.27: pulse train where signal 2 controls the width of the pulse

Remember that these all output values from 0 to 1, so in order for them to become symmetrical audio about the X axis, you should multiply the output by 2 and take 1 away to get a correct result symmetrical about the X (time) axis.

2.2.9 Cross-Synthesis

ussss.shapee

Shapee affords cross-synthesis, a sharing of frequencies and amplitudes between two sounds (normally with one influencing the frequency and the other, the amplitude). The algorithm was devised by Christopher Penrose. Versions are available for the PC by Jez Wells\(^3\) and as a plugin for Pd by Eric Lyon and Christopher Penrose under the collection FFTease\(^4\). This kind of cross-synthesis is somewhat similar to vocoding where the amplitude of one sound shapes a drone or noise depending upon the original’s spectral content. From a compositional point of view this is an excellent technique as it forms a hybrid deep

\(^3\)http://jezwells.org/Computer_music_tools.html  
\(^4\)http://www.somasa.qub.ac.uk/~elyon/LyonSoftware/MaxMSP/FFTease/
CHAPTER 2. WHAT DOES ALL THIS SOFTWARE DO?

within the sound itself. It is not just ‘A meets B’, rather something more carnal. We expose a number of tools later in this chapter that perform similar functions (see figure 2.32 and figure 2.41).

The Puredata patch

(sustains_shapes.pd)

2.3 Compound tools

Having seen the individual tools in action and having listened to the effect they have on say, the sound of a voice (a sound you know well, so something with which you can easily identify change) you are probably guessing that it is easier and quicker to plug multiple effects together in order to create a more flexible manipulation. This section looks at compound tools, where generative processes are given a second (or third) manipulation to effect a particular outcome. As disk space is cheap, you can keep most of your intermediary manipulations. As you become more proficient with these tools and develop vast quantities of sound, you may find yourself developing sounds to help you develop sounds (i.e. sounds that will never end up in the mix but which exist just to help you animate other sounds).

Let us consider the process of breaking down a sustained sound. The easiest way to break down a sustained sound is with some kind of amplitude modulation. Traditionally we have done this with envelope modulation.

(sustains_envelope.pd)

In this diagram the usss.envelope object sits in between any soundfile or granular object providing a sustained sound, and an output module. The metronome repeats a drawn envelope over a specific duration. The duration of
the envelope may roughly equal the time between triggers or it can be substan-
tially less (giving you more silence between chunks of sound).

Imagine mixing a sustained ‘Ah’ sound with a pulsed ‘a’ as in figure 2.30. Here, not only do you add a forward momentum to the original sustain but you provide the opportunity to break out of the sustain by crossfading into the pulsed version which then splutters to an exit or accelerates to something new.

![Figure 2.30: Mixing a sustained sound with a pulsed envelope](image)

A similar result could be obtained by ring modulation in sub-audio frequen-
cies. The Puredata patch

```pd
(sustains_ringmod.pd)
```

In both of these instances, the resulting envelope tends to be non-random (and may often be repetitive). If you wanted to shape the envelope of the sustained sound with the envelope of another sound, Csound’s ‘balance’ opcode could prove useful. Csound and the Java based environment for Csound called Blue will be discussed in greater length in section 2.4 on page 53.

![Figure 2.31: Envelope follower instrument in Blue](image)

When loaded into Blue, the patch

```blue
(usssustains_balance.blue)
afo1, afo2 diskin2 "<filefollow>", <speed1>, 0, <loop1>
asig1, asig2 diskin2 "<fileimpose>", <speed2>, 0, <loop2>
aenv1 follow afo1, .002
aenv2 follow afo2, .002
aenv1a tone aenv1, 100 ; smooth
aenv2a tone aenv2, 100
```
atemp1 = asig1*aenv1a
atemp2 = asig2*aenv2a
as1out balance atemp1, aenv1a
as2out balance atemp2, aenv2a
blueMixerOut "<route>", as1out, as2out

However, in cases like this we need to potentially consider designing our enveloping sound. With a sustained sound in one channel and a dynamic soundfile in the other, spectral shaping may be an option. Spectral shaping has the added bonus of not only modulating the amplitude of the source but influencing it by the frequencies shared by both soundfiles. If both sounds have strong spectral content (the sustained sound with lots of harmonically related partials and the dynamic sound being at times pitched and at other times dynamic and noisy) the shapee tool may work well.

(sustains_shapee.pd)

In this Puredata example, the output of usss.shapee goes to a delay in order to slightly spread the stereo image.

Additionally the Vocoder instrument in Blue may work well. This instrument is also quite easy to use. You do not need to worry about many of the variables as they start at reasonable default values.

![Vocoder instrument in Blue](ussssustains_vocoder.blue)

Figure 2.32: Vocoder instrument in Blue
Essentially the patch takes the frequency components of one file and the amplitude components of another and fuses them together. **Tip:** This is a great way of getting amplitude shaped materials from pitched sounds and because of the filtering effects that take place at the same time, we get additional levels of subtlety. It is always useful to have some sounds that are used only to articulate others (we use recordings of balloons and stones being heavily ‘played’). These are often granulated with lots of random ‘action’. From a compositional point of view it is good to begin thinking about files that might be good for your piece and files that might be good manipulators. If you make sounds that you know will never feature in a work, it is not necessarily wise to trash them immediately!

Returning to our initial Puredata patch for enveloping, you might consider randomising more parameters such as envelope time and granular position as in figure 2.33.

![Figure 2.33: Further randomisation of enveloping parameters](image)

The advantage of processing using multiple effects chains compared to multiple single processes (saving the file each time) is clearly a saving in time and a complexity of effect. Moreover, you begin to create the idea of an instrument in software. You may need to become familiar with sub-patching in Puredata and conditional exits in Csound (**if add_chorus=1 then goto addc**), so that you can create usable interfaces (and do make use of comments so you remember what you did previously when you return to a patch after a break away!)

### 2.4 Csound and Blue

Where Puredata (pd) allows for real-time interaction and with the **ussstoolkit**, the chance to record all your ‘play’ for future selection/deletion, Blue and Csound are a little more explicit in what they require from you. Nonetheless, we have supplied a number of quite easy-to-use tools that take the pain away from learning Csound.
2.4.1 Csound

Csound is a text-based computer programming language that has a long and chequered history. Csound began as an offshoot of one of the MUSIC languages pioneered by composer/programmers Max Mathews (1926-2011) and Barry Vercoe. With over 25 years of development, this software resource has turned into a 'real monster' with hundreds of 'opcodes' (mini-programs). As with Puredata, lots of tutorials exist and we hope that together with our code snippets and working examples you can begin to understand the real processing power of this programme.

2.4.2 Blue

Blue is a music composition environment for Csound written in Java by Stephen Yi (2008). It allows for a whole host of graphical interfaces to be incorporated into the Csound environment and brings other programming options closer to the music maker.

2.4.3 Blue examples: A soundfile player

(ussblue_playback.blue)

We begin simply by making a soundfile player based upon the diskin2 op-code. The interface shown below is found by clicking on the instrument name (sfplay) in the Orchestra tab under Interface.

![Figure 2.34: Simple playback instrument](image)

The code for this instrument is as follows. It is shown in the orchestra tab under Code:

```plaintext
; ussblue.playback
; simple playback instrument where user selects file, speed, skiptime and loop.
iflen filelen "<filein>"
istart = <skptime>*iflen
```
2.4. CSOUND AND BLUE

```plaintext
asig1, asig2 diskin2 "<filein >", <speed>, istart, <loop>
blueMixerOut "<route >", asig1*<volume>, asig2*<volume>
```

Here the skip time (where you start in the file) is given by 0 - 1 and is then multiplied by the length of the file. Select an audiofile and push play. If Blue is configured correctly then you should hear your soundfile. There is no ‘one setup fits all’ to do this: as with Puredata, if the software is not communicating with the computer’s sound card, there is some ‘under the hood’ work to be done.

Where Blue gets interesting at this level is in the Score tab.

![Figure 2.35: Adjusting playback speed](image)

Figure 2.35: Adjusting playback speed

Here we see a **Generic Score** representing a one time triggering of our instrument playing back our soundfile. Underneath is a separate layer controlling the variable `<speed>`. What this allows is a more constructed approach to soundfile creation and a more controlled shaping of transformation parameters.

**File>Render to Disk** when ready to save to your hard drive.

### 2.4.4 Blue examples: A brassage filter machine

*(usssbluebandpass.blue)*

We now jump to a different level of complexity in this example. At the outset in the orchestra tab we get a soundfile loader like before.

![Filter Sweeper](image)

Figure 2.36: another soundfile loader
CHAPTER 2. WHAT DOES ALL THIS SOFTWARE DO?

If you load up a file and then switch to the score tab you can define how long your instrument will play by adjusting the white PythonObject block. Make sure you also have green start lines at zero (left click at start on timeline) and yellow end lines after the end of your object (right click on timeline).

If you do not see a new interface you need to select **Window::SoundObject Editor** and look for the interface tab. You should then see the following:

![Figure 2.37: bandpass filter score python object builder](image)

This patch samples from the full range of your source file taking a small segment of sound (almost like a grain but a little larger, here called an element). To this element it applies a filter sweeping randomly with randomly varying bandwidth change. The number of elements taken is specified at the bottom of your randomised controls. This is almost like a granular synthesizer except that different processes are happening to each element. As the sound is filtered quite
heavily you may need to bump up the gain of the output file by normalizing to -3dB after rendering to disk.

You may ask the question: How is this happening so fast and why is there a small delay after I press play?

Have a look at the following code:

```python
import random
cfreql = <cfl>
cfreqh = <cfh>
bwidthl = <bwl>
bwidthh = <bwh>
numelements = int(<numelem>)
def myscore(numelements):
    scoreText = ""
    for i in range(numelements):
        amplitude = random.uniform(1.0, 1.0)
pitch1 = random.uniform(1.0, 1.0)
duration = random.uniform(0.05, 0.1)
skip = random.uniform(0.0, 1.0) #this is now between 0 and 1
    for full random sample time:
        cfreq1 = random.uniform(cfreql, cfreqh) #set between 50 and 10000?
        cfreq2 = random.uniform(cfreql, cfreqh)
        bwidth1 = random.uniform(bwidthl, bwidthh) #set between 100 and 2000?
        bwidth2 = random.uniform(bwidthl, bwidthh)
        scoreText += "i1 " + str(start) + " " + str(duration) + " "
        + str(pitch1) + " " + str(amplitude) + " " + str(skip) + " " + str(pitch1)
        + " " + str(cfreq1) + " " + str(cfreq2) + " " + str(bwidth1) + " " + str(bwidth2) + " "
    return scoreText
score = myscore(numelements)
```

The randomisation taking place here is being controlled by some Python code. Whilst this is rather tricky to understand it is important to note that the line `start = random.uniform(0, 20.0)` is telling Csound to place an element somewhere between zero and 20 seconds in the output file but that this is then scaled by the size of the PythonObject box.

### 2.4.5 Blue examples: A more complicated Python example: Multiple soundfile brassage

Taking this one step further the `usssbluesfmangler.blue` contains three Python functions which work on a complete folder of files. The `myscore` function is a relatively normal mixer; `articulate` applies an acceleration or deceleration to the elements and `regulator` spaces elements evenly. To enable this to work effectively you will need to have a solid understanding of the Python code and know where to find your folder of files. As the examples used in this blue patch are from a Linux machine the soundfile directories are of the form `/home/myfolder/anotherfolder/`. File paths will be structured and named differently on your computer. You might have some luck just adjusting parameters from the sample code provided. Always remember to uncomment the function call that you require (`score = myscore, articulate, regulator`).

### 2.4.6 Blue examples: A special granulator

Granulators exist in all our toolkits and Blue is no exception.
(ussbluepartikkel.blue) uses the Csound partikkel opcode. This opcode was written to explore the full range of granulation described by Roads in Microsound (Roads, 2004). We have covered up many of the variables leaving only an essential selection in the interface.

If you read Adrian Moore’s composer recollections (B.1 on page 89) you will see that there is something rather interesting about this granulation. The controls function pretty much as the granular synthesizer in Puredata.

- **speed**: speed at which you traverse through the input file.
- **grainrate**: number of grains used to create output file (giving an overall level of density of output)
- **grainsize**: length of each grain (normally between 50 and 150 ms)
- **transposition(cents)**: Transposition of resulting output file. There are 100 cents in a semitone so there is the potential for some large positive and negative numbers here
- **timeposrnd(0-1)**: A degree of jitter applied to the input pointer (where in the input file a grain is taken)
- **transprand(cents)**: randomisation of output grain transposition (0 - steady state, anything else - pitch jitter)
2.4. CSOUND AND BLUE

- **panning**: 1 for stereo panning.
- **graindistribution(0-1)**: where each grain should be placed randomly in the stereo space (this should normally verge towards 1 to give a good stereo spread).

Finally, there is an envelope that you can draw to say exactly where in the input file the granulation takes place. This is dubiously related to *speed* and *timeposrnd* so may deliver unexpected results. Some people like these unexpected results (more than others). Compare and contrast this granulation with the `uss.s.granular` in Puredata. We hope you will find noticeable differences.

### 2.4.7 Blue examples: Comb Filters

`(uss.sbluecombs.blue)` uses the Csound `vcomb` opcode in addition to a host of other functions.

![Figure 2.39: The Comb Orchestra interface](image)

The orchestra interface is simple, with a file selector and check boxes to experiment with additional filtering and envelope balancing with the original soundfile (which, as it states will lose any resonance you have acquired along the way - but an interesting modification none the less).

Subsequent to selecting a soundfile, you need to proceed to the Score and SoundObject Editor. The python code that is most important here is:

```python
direction = random.uniform(1.0, 1.0)
//
// and
//
score = combit([50, 100, 200, 300, 400, 500, 800, 1000, 2000, 5000, 8000, 10000],
1.0, 0.5, 20.0)
```

Changing the direction randomisation will give you glissandi in frequency. (0.5, 1.5) will create upward and downward glissandi. The score line is the main performance line. As you add more comb filters (specified by frequencies between the square brackets) be wary to adjust the amplitude at the end of that line. The start and end reverb times provide a nice way of moving from dry to wet.

If you like this kind of filtering then `(uss.scomobjbuild.blue)` adds a neat front end to the comb filter patch giving you control over resonance, filter frequency fundamental and number of filters (See figure 2.40).
CHAPTER 2. WHAT DOES ALL THIS SOFTWARE DO?

2.4.8 Blue examples: Convolution: a different kind of cross synthesis

Convolution in Csound is very different to the cross synthesis you have seen in Puredata. (ussblueconvolve.blue) uses the Csound convolve opcode.

In the commercial world convolution is used for reverberation. An acoustic snapshot is taken from a venue (a very short recording, called an impulse response) and used to colour all input sounds. The impulse response is the response of the building to every frequency. Therefore as you filter your input sound through this impulse response, every frequency that is in your input sound is affected by the degree to which that particular frequency was affected in the space when the impulse response was recorded. So when a noise burst is fired in the Sydney Opera House, some frequencies will decay quicker than others. It is this decay multiplication that gives you the reverb profile of the hall and therefore allows you to theoretically place your dry input sound in the Sydney Opera House (at a considerably reduced cost compared with flying to Australia).

You can create reverberation profiles of your favourite spaces by popping a balloon, hitting a snare drum, firing something legal or potentially clapping...
your hands. Take off the original sound and you have a profile. It is not that
clinical but you could then convolve your profile with your dry input recording
and add reverberation. However, this profile need not be an impulse response.
In fact it could be any small strand of sound (more acoustic DNA profiling).

The Blue convolver takes a playfile and a convolution file. You select from
where in the convolution file you wish to sample (and this is trial and error as
the start point is 0-1 representing the total duration of whatever file you load).
The slice is normally less than one second. The smaller the slice the more you
continue to hear any articulation in your playfile.

Word of Warning. It is very easy to lose high frequencies with convolution so
it might be worth considering artificially hiking up the high frequency content
of your convolution file (often to a ridiculous level). Remember we are using
this process for compositional purposes and if you get a resultant soundfile with
frequencies between 20hz and 2000hz, not only have you got one dull sound,
but there's nothing really left to process in the future. There are times when
you want this loss of quality but they are rare. So consider ramping up the
higher frequencies of your convolution file (and remember to `save as' because
you are probably creating a file that will not feature in your composition). See
figure 2.42

![Figure 2.42: a filter of some 12dB on a convolution file so as to maintain high
frequency output after convolution](image)

2.4.9 Blue examples: The Phase Vocoder - time stretching

The Phase Vocoder is a particular kind of synthesizer that works by analysing
the spectral content of your sound through tiny windows. We are now working
in the frequency-domain as opposed to the time-domain which was used for
the majority of the previous patches, especially those in Puredata. The time
stretch unit allows you to change pitch without changing speed (compare to
usss.varispeed which emulated an old analogue tape machine - slow became
low, half speed equaled octave drop in pitch, fast became high, double speed equaled octave rise in pitch) or change speed without changing pitch. When changing speed without changing pitch, you will use the time pointer (probably within the score window so you can make time varying adjustments). With the time pointer at zero you are just sitting at the current input file read point.

The phase vocoder windows can be of various sizes and we normally use 128, 256, 512, 1024, 2048, 4096, 8192. As the fft size increases the pitch analysis of the input file is increased. However, rhythmic definition is decreased as the distance between ‘slices’ has grown. This trade-off is useful and audibly very noticeable. If you want to slightly blur a file with strong articulation you might consider using a higher fft size. If you want to keep the articulation intact, make sure the fft size is 512 or less.

![Figure 2.43: A Phase Vocoder timestretch instrument](image)

2.4.10 Blue examples: The Phase Vocoder and frequency shifting/stretching/blurring

(usssbluestretchblur.blue) affords time stretching and blurring. The blur function acts as an averaging device on both frequency and amplitude. It has a very particular output sound so should be used sparingly.

(usssbluescaleshift.blue) affords frequency shifting and scaling. Here the amplitudes of component frequencies remain the same but the frequency place-holders are changed. A frequency spectrum of 100, 200, 400, 800 hz scaled by a factor of two will change to 100, 400, 800, 1600 hz. When the scale function is a fraction, harmonic input sounds often become inharmonic on output. Additionally the scale function can change over time in Blue (see the score tab). The shift function literally shifts a block of frequencies from place X to place Y. This can be good for shifting a cluster of frequencies from one region to another.
2.4.11 Blue examples: Spectral Warping

(usswarp.blue) affords experimentation with spectral warping.

The warp tool is like a more refined spectral scaler. The graph maps input frequency content to output frequency content.

(usssbluewarp2.blue) affords warping between position A and position B over the duration of the soundfile.
2.4.12 Some final thoughts on Csound

Csound is perhaps the most versatile (and well documented) language available for computer music. Whilst we have produced many examples shrouded by a Blue interface, there are countless other toolkits available. Composers at Sheffield continue to use the set developed by composer Iain McCurdy. These tools perform a number of classic and novel manipulations on sound and use the FLTK interface built from within Csound (so all you need is the latest version of Csound to get going). Iain has given us a number of very useful implementations of granulation which remains, currently, the tool of choice for many electroacoustic composers.

2.5 Supercollider

A relatively recent addition to the tools offered by USSS, Supercollider allows for some very interesting real-time manipulations especially in the frequency-domain. 

(ussfftcollider.scd) comprises many of the fft tutorials in working stereo models. Consider the following code snippet used in Click by Adrian Moore in 2011 on a granulated speech sound.

```
//*************PV_LocalMax***************/
//*****Passes only bins whose magnitude is above a threshold
//and above their nearest neighbours*******//
//PV_LocalMax(buffer , threshold)
//clearly delivering very sinusoidal residual components (the loudest bins)
//*************Very clean output*************/

( SynthDef ("PV_LocalM", { arg out\=0;
var in , chain ;
in = PlayBuf . ar (2 , b , 1 , loop : 1 ) ;
chain = FFT ( [ LocalBuf (2048) , LocalBuf (2048) ] , in ) ;
chain = PV_LocalMax ( chain , MouseX . kr (0 , 50) ) ;
Out . ar ( out , 0.5 * IFFT ( chain ) , ) }) ;
).
load (x);
 origin=/home/adrian/soundfiles/bankstreet/eyesAccountingsteady.wav
y = Synth . new("PV_LocalM") ;
y . free ;
```

Having set up your Supercollider system, make sure the line \texttt{b=Buffer.read} is referencing your own soundfile. Place the mouse on the this line and execute; Highlight from the first open bracket to the end of \texttt{.load(s)} and execute; Highlight \texttt{y = Synth.new("PV_LocalM")}; and execute. As you move the mouse your sound verges from the original to something considerably more sinusoidal. Compositionally this could quickly become clichéd but it remains an excellent means of generating a spectral decrescendo.

2.6 The three amigos: Supercollider, Csound, Puredata

Certainly, each piece of software has its own steep learning curve and, broadly speaking does similar things. However, each software has its own beauty, style,
use and community. You are very likely to find one piece of software that suits you and most of your needs. However, as you probe each tool, not only will you find the similarities quite reassuring but you will realise that there are aspects of each which will speed your workflow and potentially, bring order and structure into your transformation operations. The communities for all three pieces of software are generally very welcoming and, if you have made an effort to understand something, will often help you solve your problems.

The biggest difference between the three amigos is their level of interface with the user. Csound assumes very little (though Blue gives you an interface where you can set defaults and drivers for example). Puredata, being a graphical language naturally wants you to set up your interface beforehand. Supercollider (at least on Ubuntu Linux) assumes a great deal and has a steep curve to enable the server to work. Once working however, the program is remarkably concise and there is even a Supercollider mode that works in gedit (the basic text editor).

Take for example the creation of two sine waves. In Supercollider (at least within the Ubuntu framework with the server up and running), this is essentially one line of code. 

\[
{\text{SinOsc.ar([200, 300], 0, 0.5)}.play;}
\]

Csound demands more of the initial settings to be stipulated from the outset so the code is slightly more convoluted.

```csound
<CsoundSynthesizer>
<CsOptions>
-odac -iadc -d ; ; ; RT audio I/O
</CsOptions>
<CsInstruments>
    // Initialize the global variables.
    sr = 44100
    kmps = 44100
    nchnls = 2
    ; Instrument #1 - a basic oscillator.
    instr 1
        kcps = p4
        kamp1 = p5
        kamp2 = p6
        ifn = 1
        a1 oscil 10000, kcps, ifn
        outs al-kamp1, al-kamp2
    endin
</CsInstruments>
<CsScore>
    ; Table #1, a sine wave.
    i 1 0 16384 10 1
    i 1 0 2 200 0.0 1.0
    i 1 0 2 300 1.0 0.1
    e
</CsScore>
</CsoundSynthesizer>
```

Puredata (being graphic) is just different, though you can view pure data files as text and the patch in question is only 12 lines long (and these only describe the construction of the objects in question).
Other simplistic functions like the playback of a soundfile with mouse control over pitch involve thinking about solutions in each piece of software completely differently. Supercollider again wins the day with brevity of code and succinctness in style.

```plaintext
b = Buffer . read ( s ,"/home/adrian/soundfiles/testsounds/soundpack/speech.wav") ;

SynthDef ("p2" , { arg out=0, mysound , myenvsound , gate=1;
  mysound = PlayBuf . ar (2, b, [BufRateScale.kr(b), MouseX.kr(0.0, 2.0)], 1, startPos: 0.0, loop: 1.0);
  myenvsound = EnvGen . ar (Env . adsr , gate , doneAction : 2) * mysound;
  Out . ar (out, 0.5 * myenvsound);
  )}} . load ( s ) ;

y = Synth . new("p2") ;
y . free ;
```

Csound requires a small interface for the mouse to function as a controller.

```plaintext
<CsoundSynthesizer>
<CsOptions>
  -o dac -iadc -d ; ; ; RT audio I/O
  -r rtaudio=JACK -rtmidi=alsa hw: 1 -o dac: system: playback
</CsOptions>

<CsInstruments>
  sr   = 44100
  kr   = 44100
  kmps = 1
  nchnls = 2

  instr Process
    idur = p3
    iamp = p4
    iprd = 0.1
    ; The x values are from 1 to 30.
    ixmin = 0.0
    ixmax = 2.0
    ; The y values are from 1 to 1.
    iymin = 1
    iymax = 1
    ; The initial values for X and Y are both 15.
    iinit = 1
    iyinit = 1
    kenv . adr 0.01, 0.01, 1, 0.01
    kx, ky xin iprd, ixmin, ixmax, iymin, iymax, iinit, iyinit
    ain1, ain2
    diskin2 "/home/adrian/soundfiles/testsounds/soundpack/speech.wav", kx, 0, 1
  outs ain1 + kenv * iamp , ain2 * kenv * iamp
endin
</CsInstruments>

<CsScore>
  ; Table #1, a sine wave.
  f 1 0 16384 10 1
  ; Play Instrument #1 for 30 seconds.
  i 1 0 30 0.5
</CsScore>
</CsoundSynthesizer>
```

Puredata requires the soundfile to be loaded into an array which makes this patch very complex indeed (and not worth printing in text!)

As Supercollider has only recently been added to the curriculum at USSS, it will be some time before a more detailed set of tools are made available for this platform. That said, Supercollider is extremely powerful and as an exercise, there is nothing better than making the same process in a number of tools. The exercise reveals that there are indeed different tools for different jobs and using the right tool can save a lot of time and effort.
Finally for this chapter on toolkits, it is worth mentioning a very useful set of tools that sit alongside Csound and SuperCollider. SoX\textsuperscript{6} is cross-platform and works as a quick and effective tool for editing and processing. From the command line try the following:

- Removing silence:
  \[
  \text{sox infile.wav outfile.wav silence 1 0.1 1\% -1 0.1 1\%}
  \]

- Changing file type:
  \[
  \text{sox infile.au outfile.wav}
  \]

- Normalize:
  \[
  \text{sox infile.wav outfile.wav gain -3}
  \]

- Filtering Bandpass:
  \[
  \text{sox infile.wav outfile.wav bandpass 100 10}
  \]

- Filtering Highpass:
  \[
  \text{sox infile.wav outfile.wav highpass 100}
  \]

- Filtering Lowpass:
  \[
  \text{sox infile.wav outfile.wav lowpass 100}
  \]

- Filtering Bandreject:
  \[
  \text{sox infile.wav outfile.wav bandreject 100 10}
  \]

- Pitch bending:
  \[
  \text{sox infile.wav outfile.wav bend -o32 0,1000,2}
  \]

- Pitch shifting:
  \[
  \text{sox infile.wav outfile.wav pitch 1200}
  \]

- Pitch speed change:
  \[
  \text{sox speech.wav speechspeed.wav speed 2}
  \]

- Chorus:
  \[
  \text{sox infile.wav outfile.wav chorus 0.6 0.9 55 0.4 0.25 2 -s}
  \]

- Compaanding:
  \[
  \text{sox infile.wav outfile.wav compand 0.3,0.8 6:-70,-60,-20 -5}
  \]

- Delay:
  \[
  \text{sox infile.wav outfile.wav delay 0 0.2}
  \]

- Echo:
  \[
  \text{sox infile.wav outfile.wav echo 0.8 0.8 60 0.4}
  \]

\textsuperscript{6}http://sox.sourceforge.net/
• Amplitude Fade:
  sox infile.wav outfile.wav fade 1.0 0.0

• Flanger:
  sox infile.wav outfile.wav flanger 0 2 90 71 0.2 sin 25 quad

• Phaser:
  sox out.wav outphasor.wav phaser 0.8 0.8 3 0.4 0.5 -t

• Pad with silence:
  sox infile.wav outfilepad.wav pad 0 2

• Multiple:
  sox infile.wav outfile.wav trim 0 2 pad 0 2 reverb

• Time-stretch:
  sox infile.wav outfile.wav stretch 2

2.8 Reverberation

We have left reverberation almost until the end (though it was briefly discussed in section 2.2.7 on page 47) as for the most part it is the final transformation that you will make (and quite often you will incorporate reverberation into your mixing environment as a real-time addition). Reverberation is a natural phenomenon so correct use should sound ‘natural’ though it is often used as an extreme sustain effect. In electroacoustic music, as we want to play with dry/wet, near/far concepts, reverberation is never usually applied wholesale to a sound but is often applied over time to make it appear like a sound is drifting into the distance. When it comes to mixing, you will not have to worry about the volume of a reverb channel as a sound will be injected into it and the resulting reverb will end...when it ends. You just need to worry how to make the right wet/dry transition/balance.

2.9 Mixing

2.9.1 Setting up the reverberation channel

To complete the reverberation example above in any DAW is relatively simple. You need to add an effects channel with a reverb plug-in on it that can receive sound from what is called a pre-fade send (the opposite of which is called post-fade send by the way.) Please see figure 2.46 below from Nuendo\(^7\) that shows the necessary steps required.

\(^7\)Although Steinberg’s Nuendo is a commercial piece of software, open source alternatives exist and include Ardour. Unfortunately setting up pre and post sends is quite difficult to implement in Ardour at this time.
2.9. MIXING

2.9.2 Mixing techniques

Mixing electroacoustic music is about as different from mixing popular music as it possibly can be. Tracks will carry multiple sounds and will not be dedicated to a microphone/instrument mapping (like ‘kick’, ‘snare’, ‘guitars’ etc.) Volume curves will vary dramatically and most tracks will have a pre-fade send attached to them. See the full mix of *Click* by Adrian Moore in figure 2.47 below.

A volume track fades out (dry) as - irrespective of that volume fader, pre-fade - the sound is sent to the reverberation effects channel (whose volume remains at 0.0dB as the reverb will die away naturally).

Figure 2.46: Creating a pre-fade send for use with reverberation

[Image of a mix page for *Click* by Adrian Moore]

Figure 2.47: Mix page for *Click* by Adrian Moore

One of the key things to remember when making manipulations prior to mix-
ing electroacoustic music is that your sounds need not sound like they should exist on their own in a mix. Therefore you might make a sound that is very dark and bass heavy, or one that is very fragile, light and high-frequency. These two files might, when mixed, form the perfect canopied and rooted settings for material to be placed within the spectral space. Drones that you hear from composers like Andrew Lewis or Monty Adkins are more than likely to be composites of many smaller drones. It is quite often the case that composers seek a full bodied sound (sometimes dubiously labelled ‘phat’) directly from their manipulations. Try to imagine the potential of a sound to exist as a composite at mix time.

2.9.3 When to start mixing

There are many different answers to the question ‘when should mixing begin?’ Some may start a mix file right at the start of the composition process. Others will tend towards creating a solid bank of sources and manipulations before considering mixing. Either way, you will never have all the sounds you need prior to a mix. The generate, reflect, accumulate, mix diagram (1.2 on page 9) suggests that you will, at some point, find gaps in your mixing where creation of a specific sound file is required. It is at points like this where a really solid understanding of technique and an intimate knowledge of the sounds you have to hand are vital.

2.9.4 Where to start mixing

On a practical note, never start mixing at 0:00:00 in your DAW. Chances are you are not actually starting at the beginning of your piece so to then shift a whole bunch of sounds and effects in order to insert a new sound will give rise to errors. Start at an arbitrary time, well into your DAW’s timeline.

Consider the mix in stages or indeed, in phrases. Bear in mind too, that phrases may eventually need to be split (DAW folders help enormously here). For example see figure 2.48:
2.9. MIXING

Remixed phrases may need some additional ‘glue’ to make transitions seamless or to cover up changes in density. Critics of this style of composition have called this bricolage, as properties shared between ‘blocks’ of sound are not easily identified. The process requires some ‘fiddling’ (bricoler in French means to tinker) but as composers such as Jonty Harrison have argued, the concrete link between you and what you hear directly from the loudspeakers as you juxtapose two blocks of sound has just as much validity as two related sequences of notes on manuscript. Unfortunately, despite a vast array of powerful processing tools, when it comes to mixing we almost always appropriate the rock and roll oriented DAW where time moves from left to right and tracks are normally meant to contain one particular instrument. Frustrating moments arise when you drag in a two minute granulation file and use five seconds of it in a mix. What do you do with the remaining sound? It would be wonderful to see, when you click back on that sound in the pool, where in the mix that sound was used and what portion of it was used. There is a nagging feeling in the back of your mind as you drag in another file from a pool of two gigabytes worth of sound that perhaps this is not the sound you were meant to be using. The only way around this is to have a very solid understanding of all your soundfiles and to mix, mix and mix again. As you become more confident at mixing, consider not bothering about making sure each phrase has a start and an end; sort those out later.

Juxtaposition (with or without slight overlap) of submixes or phrases as in figure 2.48 is very different to juxtaposition of sounds to create sound objects as mentioned in section 1.4 as the time scales are so different. You are now working more at a ‘scene by scene’ level: time is moving slower. That is not to say that you can not cut away or insert dramatic changes in sound. It is that these cuts or insertions are going to form key moments in the formal design of your piece. They work best in the following two scenarios:

- Sudden dramatic insertion of attack-led phrase.
Clearly tension is easing and as this texture or drone section carries on, we are less and less sure of what to expect next. There is an air of suspense. Small sounds could appear but in this instance we have a sudden explosion of sound that should come as a complete shock. That shock is sufficient justification for it to be there: memories are yanked completely into the present as we assess the barrage of sound around us. It is only later that we can reflect upon what has just happened. This technique (someone/something approaching from behind) can be seen in films and we suspect that the first, second, maybe third time round, it is not going to be the foe approaching but a friend (a false alarm). This is harder to achieve in sound unfortunately and this process quickly loses its potency if repeated often.

- Sudden cut away to steady-state.

The cut away is perhaps even more effective than the surprise attack. When sound suddenly appears we require justification for it to happen (where has that energy come from?) The sudden cut away is like someone turning off the lights. We accept far easier the sudden loss of energy. There is an easy justification for whatever remains as a remnant or echo of the previous energy filled activity. We are still moving along at a hundred miles an hour so as we slow down we are happy to pick up on the small, inconsequential sounds left behind. The best example of this cut away is in Bernard Parmegiani’s work *Dedans Dehors* (Parmegiani, 1977) in the third movement *Retour de la forêt* as shown in the sonogram below (figure 2.51).
For over two minutes Parmegiani sustains a crescendo of epic proportions including the surreptitious mixing of a rising synthetic drone within an otherwise natural environment and a sustained tone that anchors the listener when the rest of the environment is torn away. The relief felt when another sound breaks this virtual silence is palpable.

Finally, remember overlaps? In a great deal of classical music, you often hear a phrase end overlap with the start of a new phrase. The same often applies in electroacoustic music, especially if reverberation is used as a termination feature. Where phrases start and end however, is precisely up to you. As a phrase ends, breathe and conduct where you want the next sound to start. Then make sure you place that sound at that point in your DAW mix page. Silence is used less and less in electroacoustic music as attention spans immediately dwindle but if you need.............silence.............between......phrases, use it carefully.

We are verging on a discussion of structure and form (tentatively worked through in chapter 3 (pages 81 and 83). However, from the above you can see that structure is organic: you only have be be invasive when structure starts to go wrong. Form can be both organic (intrinsic to the sounds/objects/gestures/textures you are working with) and artificial (extrinsic, the most obvious being movement form). Where structure ends and form starts is a nebulous boundary in many cases.

2.9.5 Cataloguing soundfiles

And here you might find yourself coming full circle because if you need to catalogue soundfiles for a mix, you need to be able to ‘chunk’ files into at the very least folders of some type or other. These folders may exist on your local drive or exist in your DAW profile. Similarly soundfiles will need meaningful names. Research into data management (Eaglestone et al., 2007), (Collins, 2001) (Dahan et al., 2003) continues to highlight the gap between a quick and
successful means of grasping the most pertinent qualities of a soundfile short of playing the sound and its incorporation into a mix. Machine profiling and learning has developed a great deal since researchers such as Michael Casey helped develop the MPEG-7 media content description standard (Casey, 2001) but automatic listeners embedded in your DAW are still some way off (though some specific programs such as soundfisher\textsuperscript{8} have been around for some time. Whilst a number of useful taxonomies of sound exist, the link between sound source (if recognised), any surrogate spectromorphological trace (i.e. any broad ranging descriptor of a sound), the meaning of the sound (often, by this time quite personal though not necessarily wholly subjective) and \textbf{the necessary tool to develop it} continues, for the most part, to rely upon the experience of the individual composer. This book helps (hopefully) to fast-track experience in a creative direction.

\textsuperscript{8}http://www.soundfisher.com/
Chapter 3

The theory of opposites

Student: Are there any rules here?
Teacher: Yes, the trick is to know which rules you can bend and which rules you can break.

3.1 Introduction

3.1.1 Questioning, Opposing, Arguing

In Chapter 1 you were given a whole set of descriptors that went some way to making a sound personally useful and meaningful to you, making it possible for you to describe potential development strategies and place the sound within a mix. As you begin to work with particular techniques you may find they associate themselves strongly with particular sound families. You may well be able to predict (imagine) the resulting sound’s manipulation. You may also think that something might work but have to try a ‘what if’ approach. There is absolutely nothing wrong with this as it either generates positive or negative results, both of which are usefully added to your experience.

A sound may suggest to you its acousmatic-potential by being strongly focused towards one or more particular descriptor. It tends to be the case that this potential is audible because of an abundance of said quality, not because of a lack. Therefore it would be reasonable to assume that you might want to lessen said quantity rather than amplify it when it comes to processing. This simple theory, and it should be taken as but one of a limitless number of theories that you will no doubt formulate throughout your composition career, considers that we choose transformations based upon the opposite of a sound’s most potent descriptor and react against the prevailing tendency of the sound.

In Chapter 2 we introduced a variety of tools that performed quite specific functions. In describing the processes and in listening to the results of said processes on a number of test files (and we have mentioned before that quite
often the best test file is some recorded speech\(^1\), it should have become clear that these processes also tend towards very specific results. Processes least likely to have such a polarised effect are eq and mixing. However, both eq and mixing may be used to extremes should you so wish.

It is evident too that our processes produce their best results when given the right sort of soundfile. The results of a high-pass filter or resonant/comb filter on a simple flute note for example would be negligible. Anyone familiar with what a filter does will tell you that essentially it is subtractive and therefore if it does not have anything to work on (noise based or dynamically and spectrally active material), results will be inconclusive. Similarly pitch shifting where no recognizable pitch is present produces a result that is far less convincing than if, say, the above flute note were used. Time-stretching a sound with lots of silence is futile. (Time-compression on the other hand may work). However you might well consider editing out the silence before compression or stretching, either manually or with SoX (see section 2.7 on page 67).

Let us consider a very simple series of examples:

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Process (per sound)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gritty (implies broad spectrum)</td>
<td>Filter (lessen), reverberation (dissipation)</td>
</tr>
<tr>
<td>Dull spectrum</td>
<td>Stretch spectrum</td>
</tr>
<tr>
<td>High pitch</td>
<td>Make low through pitch transposition or mix as strata</td>
</tr>
<tr>
<td>Low pitch</td>
<td>Make high through pitch transposition or mix as strata</td>
</tr>
<tr>
<td>Short</td>
<td>Make longer (reverberation, granulation, time stretching)</td>
</tr>
<tr>
<td>Discrete</td>
<td>Make continuous through repetition and granulation</td>
</tr>
<tr>
<td>Continuous</td>
<td>Make discrete through envelope shaping</td>
</tr>
<tr>
<td>Static (Mono)</td>
<td>Add motion either through panning or granulation</td>
</tr>
<tr>
<td>Spatially very dynamic</td>
<td>Make mono</td>
</tr>
</tbody>
</table>

Table 3.1: Working against single sound descriptors

\(^1\text{ftp://ftp.shef.ac.uk/pub/uni/projects/cmd/guit_and_voc/speech-test.wav}\)
3.1. INTRODUCTION

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Process (spatialisation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low bass</td>
<td>Static position in the sub woofer</td>
</tr>
<tr>
<td>High frequency, wispy material</td>
<td>Quick motion, above our heads</td>
</tr>
<tr>
<td>Thematic material</td>
<td>Generally front and centre</td>
</tr>
</tbody>
</table>

Table 3.3: Articulating spatialisation

It is very difficult to produce concrete examples of sounds that have undergone such processes but let us say we consider a simple text phrase. From the speech example mentioned earlier (from Goethe’s Faust): “To please the good old public I’ve elected, who live, and let live, them I’d recreate”. Out of context this sentence is meaningless so we should perhaps consider our first process to be to destroy any sense of word recognition. However, we will probably want to keep the sense of ‘voice’ as this is something tangible for the listener to work with.

Our options are:

- Granulate individual syllables giving us relatively smooth ‘oooo’, ‘eeeee’ and other noise-based sustains.

- Edit the file taking out first vowels, then plosives, fricatives and other vocal stops (either shortening the file or inserting silence). Granulate this file to give a slightly longer variety pack of vowel-based material or percussive / noise-based sounds.

- Working against the static pitch nature of the ‘oooo’ drone, create a number of undulations or glissandi either in a phase vocoder instrument in Blue or a simple pitch shifter (Blue or usss.varispeed).

- Working against the mid-range content of the percussive material, or perhaps more obviously, breaking the rules and assuming that you want something percussive but lighter, high-pass filter the percussive sounds leaving quite a weak, wispy version that can fly around in the space (and consider
adding randomised panning at this point). (sustains_envelope_panned.pd) would be an effective tool for this purpose. (see figure 2.33 on page 53 and consider adding a bandpass filter or hp~ object just prior to output.)

- We are mainly working in a pitch free zone. Try some delicate comb or resonant filters (see figure 2.39 on page 59) to add some pitch to our percussive sounds.

- Work against the continuous pitched files by first amplifying high frequency content then spectrally shaping against some of the percussive files. This could be done more simply by using amplitude modulation (Csound’s balance opcode) for example.

You will notice that we have come full circle from creation of sustained vowel sounds and creation of percussive textures drawn from hard consonants, to amplitude or amp/spectra modification of sustains by percussive sounds, giving rise to a hybrid of (probably) softer percussive sounds. All these new sounds are not only ripe for mixing against the original versions but are more than likely fit for further development.

Tables 3.1, 3.2 and 3.3 are just three snapshots of possible activity. As you begin to describe your sounds more accurately (perhaps beginning to feel them rather than catalogue them), and understand the potential ramifications of processing tools, so you will, for the most part continue a process of:

- sound+manipulation = sound

- +manipulation = sound

- +manipulation = sound

- and so forth.

3.2 Effects and descriptors

In very broad terms it is worth specifying the effects from chapter 2 against the more obvious poles to which they may process your sound.
3.2. EFFECTS AND DESCRIPTORS

Subtractive tools
Filters

<table>
<thead>
<tr>
<th>Effect</th>
<th>Descriptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandpass</td>
<td>Requires signal with full frequency content</td>
</tr>
<tr>
<td>Comb</td>
<td>As bandpass. Colouring filter. Rough becomes smooth as resonance increases</td>
</tr>
<tr>
<td>Reson</td>
<td>As Comb filter</td>
</tr>
<tr>
<td>FFTfilter</td>
<td>As bandpass. Noisy can become sinusoidal</td>
</tr>
<tr>
<td>Filtersweep</td>
<td>As bandpass. Spectral (time-domain) glissandi added. Flat becomes undulating</td>
</tr>
<tr>
<td>FFTBlur</td>
<td>Rough becomes smooth</td>
</tr>
<tr>
<td>FFT window size</td>
<td>Discrete becomes continuous (as ft size moves from 128 to 8192) see 2.4.9 on page 61</td>
</tr>
</tbody>
</table>

Table 3.4: Subtractive effects

Pitch and Frequency
Shifting

<table>
<thead>
<tr>
<th>Effect</th>
<th>Descriptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Varispeed</td>
<td>Low and slow tends to high and fast and vice versa</td>
</tr>
<tr>
<td>Pitchshift (Pd)</td>
<td>Speed remains the same however low to high or high to low</td>
</tr>
<tr>
<td>Phase vocoder pitch shift</td>
<td>As Pd pitch shift</td>
</tr>
<tr>
<td>Phase vocoder spectral stretch and shift</td>
<td>As Pd pitch shift though harmonic becomes inharmonic and in extreme cases, centered (together) becomes split (X partials remain around 200hz to 1000hz, all others drift upwards above 6000hz giving a root and canopy</td>
</tr>
<tr>
<td>Spectral Warp</td>
<td>Stretch and compress, harmonic to inharmonic</td>
</tr>
</tbody>
</table>

Table 3.5: Pitch based effects

Granulation and Brassage

<table>
<thead>
<tr>
<th>Effect</th>
<th>Descriptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Granular</td>
<td>See fig 1.19 on page 29. Granular normally tends towards the continuous but can be scattered or cohesive, dense or sparse, flat or undulating</td>
</tr>
<tr>
<td>Brassage (Blue filter sweeper)</td>
<td>Scattered and undulating with focus upon repetition (but not at granular level)</td>
</tr>
</tbody>
</table>

Table 3.6: Granulation and Brassage
CHAPTER 3. THE THEORY OF OPPOSITES

Spatialisation and delay based effects

<table>
<thead>
<tr>
<th>Description</th>
<th>Descriptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reverberation</td>
<td>Discrete becomes continuous. Near becomes far (or if wet and dry are mixed, you can get a full depth of field with frontal image and distant reverberation). Dry becomes wet. Strong becomes weak.</td>
</tr>
<tr>
<td>Delays (del time &gt; 0.5s)</td>
<td>Single becomes multiple</td>
</tr>
<tr>
<td>Delays (del time &lt; 0.5s)</td>
<td>Discrete becomes continuous (as delay and original merge)</td>
</tr>
<tr>
<td>Delays with feedback</td>
<td>Discrete becomes continuous as object receives ‘delay-tail’ as feedback dissipates.</td>
</tr>
<tr>
<td>Panning</td>
<td>Focused becomes spread. Cohesive becomes scattered. (See figure 2.19 on page 46)</td>
</tr>
</tbody>
</table>

Table 3.7: Spatialisation and delay based effects

Cross synthesis

<table>
<thead>
<tr>
<th>Description</th>
<th>Descriptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shapee and Blue Vocoder</td>
<td>Continuous becomes discrete as you normally cross synthesize a drone with articulated material.</td>
</tr>
<tr>
<td>Convolution</td>
<td>Discrete becomes continuous the longer the convolution ‘slice’. Rough becomes smooth as convolution is like a reverb or colour filter.</td>
</tr>
</tbody>
</table>

Table 3.8: Cross synthesis effects

Clearly it is never a certainty that the effect matching the opposite of sound A’s descriptors will deliver good results. You may well have to take your sound and physically move it towards one pole (or many) by using a process quite aggressively, so that sound A actually gets ‘into the system’ - a ‘what if’ approach. Although this sounds like cheating you will have recognised acousmatic-potential in a sound. Remember, a sound that sounds like the process that made it has lost a significant amount of its own acousmatic-potential but has taken on a completely new life. Listeners well versed in electroacoustic music seem happy to accept eq, mixing and granulation alongside quite a few spectral manipulations as part of the everyday toolkit. Comb and resonant filters still remain somewhat clichéd however. You will note too that only a very few of the descriptors from tables 1.1 and 1.2 are used in the examples 3.4 through 3.8. Clearly the effects used can be nuanced to a much greater degree as you become more familiar with their operation.
3.3. STRUCTURE

3.2.1 The theory of opposites

It may be useful to think about a sound’s descriptors and about what processes work well against those descriptors. From a compositional point of view, as we tend to think less about the sound itself, less too about the process and more about abstract concepts or personal emotions, theory and technical manipulation become secondary tools, serving the needs of our composition. It is a lofty aspiration but it ties neatly in with our desire to move away from ‘chunking’ sounds/phrases (in any music) towards an holistic acousmatic and subjective understanding of the structure, form and meaning of a work.

3.3 Structure

3.3.1 Smalley’s use of structure

And it is here that the theory of opposites ends (though find me a work where ‘fast’ has not eventually been offset by ‘slow’) and we enter a much more natural world of pace and design at a much larger scale than the sound or sound object. In this book the actual concept of structure begins at the mix despite the fact that you are bound to be thinking about structure while making sound. Do not be confused by authors talking about a sound’s structure. This is but a minute analytical rendition of a sound’s shape. We suggest in this book that structure is organic; that it results from the growth of sounds through objects, gestures, textures, landscapes and environments. Denis Smalley’s brief discussion of structural relationships in Spectro-morphology and structuring processes (Smalley, 1986, 88-89) is dealt with in a vastly different manner in the revision in 1997, Spectromorphology: explaining sound-shapes (Smalley, 1997, 114).

In 1986, his diagram of structural relationships seems to suggest time units larger than that of sound or sound-object. Additionally there is a noticeable use of polar opposites such as vicissitude versus displacement.

---

2Vicissitude is the continuous development of sound through morphology; displacement is a rougher transitional phase.
CHAPTER 3. THE THEORY OF OPPOSITES

Figure 3.1: Denis Smalley’s diagram of structural relationships (Smalley, 1986, 88)

In the 1997 re-working, terminology from the development of the object (see section 1.4 on page 22) is explained at larger time levels. The key terms are onsets, continuants and terminations.

<table>
<thead>
<tr>
<th>onsets</th>
<th>continuants</th>
<th>terminations</th>
</tr>
</thead>
<tbody>
<tr>
<td>departure</td>
<td>passage</td>
<td>arrival</td>
</tr>
<tr>
<td>emergence</td>
<td>transition</td>
<td>disappearance</td>
</tr>
<tr>
<td>anacrusis</td>
<td>prolongation</td>
<td>closure</td>
</tr>
<tr>
<td>attack</td>
<td>maintenance</td>
<td>release</td>
</tr>
<tr>
<td>upbeat</td>
<td>statement</td>
<td>resolution</td>
</tr>
<tr>
<td>downbeat</td>
<td></td>
<td>plane</td>
</tr>
</tbody>
</table>

Table 3.9: Structural functions (Smalley, 1997, 115)

Again, we see the continuum fully in action from pole to middle-ground to pole. And these structural functions may operate at the sound, sound-object, phrase, section and completed work level.

3.3.2 Other structure(s)

It may also be useful to think about the structure of other large scale objects such as buildings. Components here for all manner of shapes and sizes of building include foundations, walls, bricks, mortar, windows, roofs. It is a simple analogy to the above but you may well, at some point in your composition find yourself
with two phrases which need to go together but which require a transition, a kind of ‘glue’ if you like. You could also see the foundation and roof as floors and canopies with bricks, windows being grains or larger objects. The point here is that a multitude of parts are stuck together to complete the whole and one would not work effectively without the other. Whilst we have not mentioned the word time, it pervades everything we do. What is perhaps most interesting about the way you and others comprehend your work is that during the piece, comprehension is bound by the timeframe of the work but during it, before it and after it, we may assess the work outside of this framework.

3.3.3 Beginning, Middle, End

Although the techniques in chapter 2 can be applied to all manner of creative outputs like interactive pieces, installation art, audio-visual works and the like, this book’s aesthetic is directed towards the completion of a self-standing work that starts at the beginning and ends at the end, going somewhere via the middle. Get this recipe right and you are 90 per cent there. But this is form right?

3.4 Form

Eventually, after listening to a complete work you may come away with an understanding of how the whole thing works. This may be your gut reaction that you loved it, hated it, or that it requires further attention. It may also be a representation in your mind of a very high-level abstraction of the work. The easiest examples (and we are using very simple renditions here) are Andrew Lewis’ Scherzo, (1992 rev. 1993) and Alistair MacDonald’s Equivalence, (2007).

- Scherzo. Dense gestural ‘clutter’ clearing to technical mastery of vocal manipulations with a very specific narrative, diffusing to seashore sound recording.

- Equivalence. Bricks and granulation. Increasingly dense towards the middle. A difficult piece to listen to but if you familiar with granulation, you’ll relish the attention to detail and the almost impenetrable levels of mixing and eq.

Both viewpoints are naïve in the extreme and should be far more detailed. However, the point about a form in electroacoustic music is that you can rush through the piece successfully at fast-forward speeds and get a sense of the work. It is not just that Jonty Harrison’s Klang,(1982) is ‘an earthenware casserole piece’ but that its introductions, developments and recapitulation (start, middle, (Lewis, 1993)

4 (MacDonald, 2007)

5 (Harrison, 1982)
end) are of scales that are manageable. The start is gentle, the developments become quite forceful and the consequent relaxation is precisely what we hoped for.

Some forms are quite obvious. *Dedans Dehors*, (Parmegiani, 1977) was in five movements over some 25 minutes. If the work is short, it may have but one kind of pacing and style. However, after a certain duration, (and this remains one of those ‘undefined’ numbers) your piece probably will have sections. After periods of intense activity it is often the case that there are moments of tranquillity and vice versa. The coherent placement and timing of fast and slow is *structure*; the perception of different pace over a space of time in a certain proportion is *form*. Form is perceptual not conceptual (‘form first’ will only ever lead to disaster). Structure is constructional and compositional and can be perceptual too but where structure has worked when listening, we often find ourselves moving forwards in a piece. We might unpick the structure later on but we are almost working with the form at that point.

### 3.4.1 Does this matter?

Given that everything in this chapter is sound driven:

- great sounds, great manipulations, dynamic sound objects, fluid phrases, tight structure, tangible form. Excellent.

- poorly recorded sounds, very little transformation, no sound objects, some phrases, loose structure, clear ABA form based upon a direct repetition of sound. Terrible.

### 3.5 Originality

Most electroacoustic composers have used water sounds in their pieces. We have a strong affinity with water as it tends to be either exhilarating (rapids, water falls) or intensely peaceful (streams, brooks, seashore). There is a beautiful spectral ‘slurp’ in closely recorded water and exquisite ‘twinkling’ pitches. Strong metaphor too in the ebb and flow of the tides and the tension/release of waves. There is no reason not to use water in a work. But how to make your work original? There is one very easy justification: record your own water sounds. They are immediately personal to you and your playback of these sounds comes with the guarantee that ‘you were there’. It is immediately easier too, to be able to re-trigger the emotions felt as you recorded the water and amplify, extend or modify these by processing and mixing a work.

As mentioned in Alistair MacDonald’s composer recollections (see appendix B.2 on page 90), make sure you play your work to other people. You need to be there in the room too. That way you suddenly start to listen in the third person (as though you were the other person). Do not assume your piece works. Make something then deliberately break it up and see if you can not glue it back together in a different order. Finally, you must listen to music by other
3.6. A VERY OBVIOUS CONCLUSION

composers. It is extremely easy to find music on the internet (soundcloud, myspace, sonus.ca are three places to start) and many works are available on Compact Disc and DVDA (Empreintes DIGITALes http://www.electrocd.com/ is a good place to start and clips of most recordings are available). The words electroacoustic and acousmatic have been used freely throughout this book. Look them up in wikipedia and peruse the history of the genre. Whilst acousmatic music and more rhythmically driven music have merged since the start of the twenty-first century adapting to the changing nature of performance and dissemination of sound art, please consider rhythmically driven music a subset (and a highly focused subset too) of free sound art. As mentioned in section 1.8.2 on page 32 we have a strong affinity to the natural-ness (or naturality) in sound. Pitch therefore (the harmonic structuring of partials) is clearly something we can easily latch on to. Rhythm and pulse suggest machines (which, for the most part we detest) but pulse in music is ubiquitous and alludes to the pulse of life and the dance.

Nobody is going to expect to get up and dance to your experimental water-based sonic art (you never know!) but even streams, rivers and dripping taps have a natural pulse. Breaking into and out of full-on pulse is as hard as leaving a solid steady-state drone. Given that a standard electroacoustic piece can vary in duration from anything between three to thirty minutes, pitch and rhythm focus may well be devices that carve out the form of your work. It is very easy to create a pulse-based lattice and not have regular repetitions. You maintain drive without suggesting a clinical structure. With quasi-pulsed granulation it is possible to obtain a sense of pulse whilst changing many other parameters. In *Dreaming of the Dawn* by Adrian Moore (2004), there is a passage that, in the programme notes for the work, is likened to ‘the interplay of clutch, accelerator, gear shift: we are propelled forwards but there is a constant acceleration, deceleration as we move up and down the gears’.

### 3.6 A very obvious conclusion

If you have some recorded sound, have taken some time to read but a fraction of this book and have experimented with any of the software mentioned in chapter 2, unless you have given up, you might be enjoying your ability to shape sound (and time). Composition is an ongoing, lifelong process and barring expensive microphones, sound recording devices, loudspeakers and computers, can be absolutely cost free. We hope you continue to enjoy composing, continue to develop your compositional style and we ask that you make sure you communicate this pleasure, through pieces, performances, social networks, emails, books or conversation to any and all interested parties. Have fun.
Appendix A

USSS toolkits

A.1 A1. Downloading and installing software

This book uses four primary pieces of software to develop sounds, all of which are cross-platform and freely available:

- Pure Data (http://puredata.info/). The usss toolkits use Pd-extended.
- Csound (http://csound.sourceforge.net/) which is required when you use
- Blue (http://blue.kunst musik.com/), a fully programmable composition environment for Csound.
- Supercollider (http://supercollider.sourceforge.net/)

Additionally we have used two cross-platform audio editors / DAWs:

- Audacity (http://audacity.sourceforge.net/). A very solid two channel editor with multi-channel capability.
- Ardour (http://ardour.org/). A more powerful multi-channel DAW.

The usss toolkits that work with this book can be downloaded from http://www.shef.ac.uk/usss. Patches come without any guarantees and are licensed under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. (http://creativecommons.org/licenses/by-nc-sa/3.0/)
...But I am giving away my secrets!
...Nothing is secret for long.

B.1 Adrian Moore

I have tried almost every technique under the sun and if I had one piece of sound advice it would be that as soon as you think you have happened upon a sound this is particular to you (to the best of your knowledge this is currently your 'signature' sound), write down how you made it. Despite the supposed rigour of creating tool-kits in all manner of software, when it comes to manipulation, we as composers are often switching between software and soundfiles at a rapid pace, and, sometimes, forgetting to note down what went with what. I remember (or should I say, failed to remember) how I made the opening sounds in my work *Dreaming of the Dawn*. There was a strange foldover sound to them. However, I have since found a method for creating drone based material using the Partikkel opcode in Csound/Blue. Again, I’m not quite sure exactly why the transpositions are such that my drones have quite strong full spectrum colour, especially in the upper ranges (I always tend to roll off the frequencies above 15Khz by up to 12db) but these drones interest me enormously and I have used them in recent pieces such as *Tapeworm* and *Click*.

Additionally I absolutely love using convolution and spectral cross-synthesis in whatever language (convolution and vocoding in Csound/Blue, cross-synthesis in Max/MSP or Puredata using the Shapee plugin). Ever since I used convolution and cross-synthesis in *Junky* these techniques have been essential methods in all my works. Simply put, cross-synthesis fuses sound A and sound B, creating a hybrid. If sound A is pitched and sound B is articulated, suddenly your pitched sound is not only articulated dynamically but is spectrally filtered through sound B. In one easy step, you have used filters and amplitude modulation and given life to an otherwise static drone.

Adrian Moore is a composer based in Sheffield. a.j.moore@shef.ac.uk
APPENDIX B. COMPOSER RECOLLECTIONS

B.2 Alistair MacDonald

Top tips

- Listen with somebody else in the studio (it always sounds different).

- Think about space and dynamics for each sound from the start (do not add dynamics and spatialisation later - they are part of the identity and character of each sound).

- Think about energy and move around the studio when you are listening (does the music move you?).

- Think about shape - you may need to add simple layers to underpin/orchestratre the overall shape that the surface layer is articulating.

- Think about eq, a common but very powerful tool. Consider using eq on every sound, partly just to see what is there (or not there).

- Editing is probably your most powerful tool - isolate bits of your sound; de-contextualising bits will tell you things you did not know.

Alistair MacDonald is a composer based in Glasgow. a.macdonald@rcs.ac.uk

B.3 Andrew Lewis

B.3.1 First I find, then I seek

I try not to be too intentional too soon. In choosing and recording sounds, and in transforming them, I aim to have a completely open mind, and just go with the flow. I see what I can find, almost by accident, without worrying too much what I am going to do with it. I stumble across things, and allow myself to be surprised by unexpected revelations. This is ‘finding’. I try not to compose at this stage, except in the sense of putting things together to make individual objects or phrases. If I do try composing too early, it is usually very bad and very frustrating.

Then comes ‘seeking’: this means I start actively looking for specific things, trying to realise certain kinds of ideas, exploring and developing the latent possibilities of the stuff that I have ‘found’. This applies to transformation, but is also when the real composing begins, in the sense of trying to shape a whole piece. Of course, ‘finding’ can also continue in this stage, as unexpected possibilities present themselves. If they do, I try to be prepared to abandon my previous plans, and even material that I spent a long time on, and instead let this new ‘finding’ take my ‘seeking’ in a new direction.
B.3.2 The trash can is your friend

I find that it is important to reject material, both at the development stage and when composing. In the same way, muting tracks in a mix or regions (segments of sound) can be a revelation. Sometimes I have made the texture so think that really beautiful things are hidden away. Thinning things out a bit often produces surprisingly good results.

B.3.3 Listening

I agree with Alistair Macdonald about listening with someone else in the studio. It really is a revelation (usually a negative one, because you stop being able to kid yourself that ‘it’s ok really’!)

I try not to listen too much. My ears become quickly jaded, I try not to have processes droning on while I am working on something else. I also don’t listen to large chunks of the piece too often, but treat it as something special. The first listen through to things on entering the studio is crucial - I can hear everything at its clearest, both sonically and musically.

On a related point, I do not play around too much with realtime processes without recording them. If it sounds good I record straight away, otherwise I find I end up playing around with different possibilities for an hour, at the end of which I (a) have no results, and (b) cannot really hear properly any more, or decide what’s good or what’s bad.

B.3.4 Time

Acousmatic music is MUSIC, not just ‘sound art’, and the main distinction here is its use of time. It is not just that the material is ‘time-based’, but that the artistic ideas themselves are shaped by and shape time. This is different to most gallery-based ‘sound art’, where the material is still time-based, but the overall artistic conception is not - it doesn’t make too much difference in what order you hear things, or at what point you start and stop listening. In acousmatic MUSIC you must start listening at the beginning and stop at the end, and it makes a huge difference in what order things happen, and how quickly they follow. I spend a lot of time changing the order of events and their pacing, without changing the sounds themselves. To me, ordering and pacing is as transformational to sounds a processing or dynamics. (Another way of thinking about this is that I am trying to shape the listener’s perception: I put sound b after sound a, because sound b will take on certain special qualities if I lead the listener through sound a first, and thus create a particular perceptive space. This involves things like expectation, tension, fulfilment, contrast, and so on. These are all changed by different orderings and pacings of material.

Andrew Lewis is a composer based in Bangor. andrewlewiselectro@gmail.com
Appendix C

Binary Representations

Binary is the ‘language’ of the computer or electronic gizmo. It is used for the storage and processing of everything that is inside the computer. We can use binary codes to represent any data structure we like but in the studio we commonly see binary representations of audio signals, control data and measurements. A single binary digit (bit) is represented by the state of one electronic switch inside the computer. Bits can be grouped together to form a ‘word’, word lengths of 8, 16, 32 and 64 are common. For example, we might talk about a 64bit CPU, this would be a CPU that uses a 64bit word length for the majority of its operations. 8 bits is 1 byte (and more amusingly 4 bits is a ‘nibble’).

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Number of bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kilobyte</td>
<td>KB</td>
<td>$10^3$</td>
</tr>
<tr>
<td>Megabyte</td>
<td>MB</td>
<td>$10^6$</td>
</tr>
<tr>
<td>Gigabyte</td>
<td>GB</td>
<td>$10^9$</td>
</tr>
<tr>
<td>Terabyte</td>
<td>TB</td>
<td>$10^{12}$</td>
</tr>
</tbody>
</table>

Table C.1: Names for Multiple Bytes

In audio and video applications we often deal with very large numbers of bytes and single projects often require gigabytes of data storage. So how can we represent numbers? Are the numbers whole numbers (integer) or fractional (floating point)? Are they signed, both -ve and +ve. How do we store text or even audio?

C.1 Decimal (base 10)

Let us take a look back at the number system we are all familiar with. We are used to working with numbers in base ten, the decimal number system. In decimal, each digit represents the number of ones, tens, hundreds, thousands and so on. Each digit column based of successive powers of ten from right to left:
APPENDIX C. BINARY REPRESENTATIONS

<table>
<thead>
<tr>
<th>10^4</th>
<th>10^3</th>
<th>10^2</th>
<th>10^1</th>
<th>10^0</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>1000</td>
<td>100</td>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>

Table C.2: Decimal column values

In decimal we have ten symbols to represent column values and we know these as the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Large values are represented in decimal by placing the digits in appropriate columns, e.g. 345 three hundred and forty five. You remember your early school maths? ‘three hundreds plus four tens plus five units’.

\[3 \times 100 + 4 \times 10 + 5 \times 1 = 345\]

Note that a column value is given by the base raised to the power of the column index starting from zero, i.e. the thousands column is three to the left from the units column (zero index column) and we see that \(10^3 = 1000\).

C.2 Binary (base 2)

As with decimal, the columns represent multipliers but in this case they are based upon successive powers of two from right to left:

\[2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0\]

| 32 | 16 | 8 | 4 | 2 | 1 |

Table C.3: Binary column values

A simple way to remember the columns here is to start with the units column and repeatedly multiply by two until you have the desired number of columns.

C.3 Counting in binary

The following table shows the numbers 0-9 translated to binary equivalents:
C.4 Bits, Bytes and leading zeros

The number of digits in a computer representation of a binary number is the number of bits. When we specify a number using certain number of bits we fill the all the bits with 0 or 1 so \(3_{10}\) is 00000011 in 8 bit binary. Modern computers typically use 32 bit or 64 bit representations i.e. \(3_{10}\) would be 00000000000000000000000000000011 with all the leading zeros filled in.

C.5 Adding binary numbers

In the following example we can see how the same methods for adding numbers column by column work equally well for binary numbers:

\[
\begin{array}{c|c|c}
\text{Binary} & \text{Decimal} \\
101_2 & 5_{10} \\
001_2 & 1_{10} \\
110_2 & 6_{10} \\
\end{array}
\]

Table C.5: Adding binary numbers

C.6 Representing negative numbers

By using one of our bits to specify positive or negative we can represent negative numbers. The method used in computing is called the Two’s-complement. To go from a positive number to a negative we invert all the bits and add one.

C.7 Fractional numbers

In decimal the digits after the decimal place (.) are -ve powers of 10:
APPENDIX C. BINARY REPRESENTATIONS

\[
\begin{array}{cccccccc}
10^4 & 10^3 & 10^2 & 10^1 & 10^0 & 10^{-1} & 10^{-2} & 10^{-3} \\
10000 & 1000 & 100 & 10 & 1 & 1/10 & 1/100 & 1/1000
\end{array}
\]

Table C.6: Decimal column values extended

In binary we can do the same with 2s

\[
\begin{array}{cccccccc}
2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 & 2^{-1} & 2^{-2} & 2^{-3} \\
32 & 16 & 8 & 4 & 2 & 1 & 1/2 & 1/4 & 1/8
\end{array}
\]

Table C.7: Binary column values extended

C.8 Floating point

Floating point numbers are essentially a way of representing extremely large range of numbers with a smaller numbers of bits and a ‘floating’ point. You have almost certainly done this in decimal when you used scientific numbers. $0.001 = 1.0 \times 10^{-3}$ In this case the mantissa is one and the exponent is negative 3. In binary we use a base two representation for the mantissa and exponent. $0.125 = 1/8$ in decimal and $1.0 \times 2^{-3}$ in binary.

Different systems and software may use different numbers of bits for the mantissa and exponent. For example, we could use choose to use four bits for mantissa and four for exponent in an eight bit representation. One eighth would therefore have a four bit unsigned mantissa value of $1_{10} 0001_2$ and a four bit unsigned exponent value of $3_{10} 0011_2$ stored in eight bits by concatenating the bits together as 00010011. Of course it would mean something totally different depending on your interpretation of the columns.

In computing, the interpretation of bits changes according to context. Bits can represent numbers but equally they could represent commands, text or the status of a device.

C.9 ASCII Strings

Letters and other characters can be represented using a common code. ASCII character code uses 8 bit unsigned values to represent 256 distinct characters. By storing a sequence of ASCII values we can represent a sentence. This is commonly called a text ‘string’.

C.10 Hexadecimal

Hexadecimal (base 16) is a useful shorthand for binary. A sequence of 4 binary bits can be represented as a single hex digit so a single byte can be a pair of
C.10. HEXADECIMAL

Hex digits. We only have 10 numeral symbols so we extend our symbol set with letters:

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td>A</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td>B</td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td>C</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td>D</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td>E</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>F</td>
</tr>
</tbody>
</table>

Table C.8: Hexadecimal numbers represented in decimal and binary

Hex notation is used extensively in MIDI syntax descriptions. eg. SYSEX F7 and F0 status bytes

We also see it in representations of RGB/RGBA colour where we often see 3 or 4 bytes used to represent red, green, blue and alpha (transparency) components of a colour, eg. #FFFFFF is white #000000 is black #FF0000 is red
Appendix D

Puredata (Pd)

pd is a real-time graphical programming environment for audio and graphical processing. It resembles the MaxMSP system but is much simpler and more portable. For some time it sported a number of features not present in Max/MSP. With Mark Dank’s GEM package, Pd could be used for simultaneous computer animation and computer audio. Additionally, an experimental facility was provided for defining and accessing data structures. Max/MSP has subsequently incorporated Jitter and Live and has grown considerably in size, in part due to its commercialisation.

Pd is extremely useful for creating signal processing tools that operate in real-time. Its graphic interface and flow diagram programming model make it relatively simple to learn and very intuitive if you are familiar with modular synthesizers. The flow diagram concept makes it very straightforward to visualize data and signal flow through your programs.

Perhaps its biggest disadvantages are apparent when you need to process complex structures of data in a very iterative way. In these situations other languages offer better control. Pd is best suited to real-time control and live performance and is certainly not the best tool for offline or batch processing of audio/midi data.

When we first load Pd we are presented with the main window of Pd. This window is the main area for creating new Pd files and setting up the devices that Pd will use. It also displays error messages and any information that you choose to send it.
APPENDIX D. PUREDATA (PD)

In order to create programs with Pd we create patches. A patch is a like a blank canvas for you to add objects. Objects that you add to a patch can be connected together in different ways in order to describe the way that data will flow. This is somewhat akin to a flow chart or the physical wires between devices.

Pd responds to user or device generated information. These ‘Events’ are such things as mouse clicks, key presses, incoming MIDI messages, OSC messages etc. The various GUI object boxes and objects that represent devices generate messages. Other object boxes process and sometimes generate new messages. The messages flow through the diagram and this is how we create programs. This processing model is therefore ‘event based’.

D.1 Basic units of programming Pd

Object boxes either do things to data, get data from physical devices or send data to physical devices. Objects are the main building blocks of programs in Pd. The available object classes are extensive and can be extended through libraries. If you are familiar with MaxMSP there is a particularly useful extension library called Cyclone which provides clones of many MaxMSP objects. The Pd extended version comprises many such libraries.
**Message boxes** let you specify commands or provide data to objects. It is very important to learn the difference between objects and messages. In essence: objects do things and messages provide data and control commands to objects.

**Comment boxes** let you make notes for yourself and others. It is a very good idea to comment complex sections of patches that you make.

**Number boxes** allow you to enter numbers with the mouse and keyboard. They are one of the GUI objects in Pd and they can be used to display numerical information as well as enter it.

### D.1.1 Data types

**integers** (int, i) : are whole numbers. In Pd (unlike MaxMSP), there is no specific integer data type, every thing is in fact a floating point. However, its useful to understand that many objects will expect whole numbers and may round values.

**floating point** (float,f) : are real numbers. Pd uses a 32 bit floating point representation. The range is huge but as with any floating point representation precision is lost at extremely large or small values.

**symbols** (symbol) : (in other languages these are normally called strings). You often define these in message boxes as a block of text surrounded by double quotes (“a bit like this”).

**bangs** : In Pd there is a special data type that often is used to trigger the operation of an object. When you send a ‘bang’ to an object it performs its programmed task.

**lists** : A list is defined as a collection of any of the above data types. There are special objects for constructing lists but you can also define them directly in message boxes. For example a message box containing (hello world 10 20 30 40.4 bang) would construct a list with “hello”, ”world”, 10, 20, 30, 40.4, bang.

### D.1.2 Numbers and range calculations

In Pd and any computer programming language, everything eventually boils down to numbers. One of the most useful skills in Pd is to understand the ranges of the numbers generated by objects and devices. Here is a quick list of common number ranges:

- MIDI Notes: (0 .. 127) (7 bit integer)
- MIDI Velocity: (0 .. 127) (7 bit integer)
- MIDI CC: (0 .. 127) (7 bit integer)
• MIDI Pitch Wheel: (-8192 .. 8192) (14 bit integer)

• Input range of Audio: (-1.0 .. 1.0) (32 bit floating point)

• Output range of Audio: (-1.0 .. 1.0) (32 bit floating point)

• Scaling audio: (0.0 .. 1.0) (32bit floating point)

• Oscillator frequency: (0.0 .. 20000.0) (32 bit floating point) This is the range of human hearing.

• ASCII character: (0, 255) (8 bit integer representing a letter, space or symbol in an email)

Learning how to manipulate numbers from one range to another is a key skill in Pd programing.

### D.1.3 Displaying values and messages using print

The print object is one the most useful objects in the Pd toolbox because it allows you to display the contents of messages. It can be connected to the output of an object and enables ‘debugging’. Any messages arriving at the print object’s inlet will be displayed in the Pd window. Specifying a name as the first argument (the first word after print) will cause messages to be prefixed with this name. This is useful to identify which particular print object is displaying a message. As a simple example the following patch connects a number of different basic objects to the print object. Clicking on objects will cause some output in the Pd console.

Here we see the main window output that was produced by clicking on a number of different object boxes connected to a print object box.

![Figure D.2: Pd message output](image)
D.1.4 Range mapping

In order to map a number range from one range to another we have a number of mathematical methods. Perhaps the simplest is to scale and offset using the multiplication object and addition object.

By setting the scaling factor and offset values appropriately we can perform a linear remapping of an input range. A simple method for understanding this range mapping is to perform the scale and offset on the minimum and maximum of your input range. So in the example our minimum is 0 therefore output minimum is $0 \times 10 + 1 = 1$. Our maximum is 10 so $10 \times 10 + 1 = 101$. This process is often also used in signal processing using $\text{\textasciicircum}$ and $\text{+}$ objects.

Often we need to perform a more complicated mathematical operation. The $\text{expr}$ object allows complex mathematical equations to be expressed and performed on numerical messages. The $\text{expr}$ object is very powerful and includes mathematical functions such as $\text{sin}$, $\text{cos}$ and $\text{pow}$ (raise to the nth power). This patch calculates the magnitude of a vector specified by two numbers and highlights some of the required syntax.

There are also a number of musical range conversion objects that are particularly helpful. $\text{mtof}$ converts from a MIDI note value in the range 0..127 to a frequency in Hz. This object is useful for creating traditional synths and for mapping linear controllers to a logarithmic frequency scales. Its inverse, $\text{ftom}$,
can be useful when converting analysed audio signals into MIDI. This example converts an incoming note number from a MIDI device to a frequency in Hz and uses the frequency to control the pitch of a sinusoidal oscillator.

![Pd mtof example](image.png)

Figure D.5: Pd mtof example

D.1.5 Getting numerical data from MIDI controllers and keys

We can use the the notein, ctlin and key objects to grab incoming values from MIDI devices and the keyboard. The following patch shows how we can visualize the output using the print object.

With a combination of simple range mapping and some physical input we can start to make a real-time musical tool quite quickly. This example shows how we can use a simple range adjustment to map a physical slider into a range suitable for controlling the bpm of a metronome. A bleep is synthesized with a simple envelope generator line~ and signal multiplication *~.
D.1.6 Lists

The list data type is very important for controlling objects. It is vital to be able to construct and deconstruct lists of data using the objects that Pd provides. Lists can be manually entered into message boxes. In a message box the space character is used as the list element separator. We can use a number of special identifiers in a message box to insert data from an inlet into a message. If the message is formatted as a list then the output generated is a list message.

An alternative method of creating a list is to use the pack object. This object allows you to define the data types and number of elements to be packed together.

In order to deconstruct a list into its constituent elements we can use the unpack object. The following example constructs lists and then unpacks them further down the patch:

![Diagram of a Pd metronome using range scaling]

Figure D.6: Pd metronome using range scaling
D.1.7 Sending and receiving data without connections

Objects and data in Pd can be sent directly to objects without the need for connections. In fact, data can be sent from one patch to another if two are loaded at the same time. The send objects allows data to be sent to named receive object. You can have multiple receive objects and multiple send objects with the same names and all of them will intercommunicate. In addition most GUI objects are able to send and receive data directly to named locations. This example shows some possibilities. Note the use of (r and s) shorthand for send and receive.
D.1.8 Tips for exploring Pd

- Go through the tutorials.
- Use the browser, section 1 is the manual for Pd.
- Use the help pages for objects (right click and select help). It’s worth noting that the help is both interactive and editable so you can cannibalize help patches in order to understand them.
When reading the help page the ‘see also’ section is also very useful for finding other interesting objects. Information is always given about the type and expected data ranges for each inlet and outlet. Details about creation arguments and optional arguments are also present. Try to identify the following keywords in the syntax information in order to understand an object:

- **inlets**: the messages and data expected at inlets, and what is done with it.

- **outlets**: the messages and data generated at outlets. What could you use the data for? What do you need to send in order for the data to be generated?

- **arguments**: the data that you put next to the name of the object. Arguments are creation time parameters only. Could you use an argument to specify something without needing to connect another box? Some objects require arguments in order to be created.

## D.2 Signal Processing in Pd

### D.2.1 The audio signal data type

In Pd, audio signals are processed differently from events. An event is processed on demand: the event happens, Pd processes it then waits for more events. Signals are computed continuously once the ‘compute audio’ option is set. Real-time, host based signal processing occurs in blocks and the size of the blocks is called the vector size or buffer size. A small vector size will allow the system to respond very quickly and a larger vector size will require the processor to wait for data and then process it all at once.

Our knowledge of sampling theory tells us that a signal is really a sequence of discrete samples. In Pd each sample is stored as 32 bit floating point numbers and the complete dynamic range of the audio device is mapped over the range -1.0 to 1.0.

Processing of signals only happens when the DSP system is enabled, the compute audio option on the Pd main window allows enabling and disabling. However, there are other ways to do this directly from a patch. A special message box containing `; start dsp` or `; stop dsp` will do a similar job. This patch demonstrates some of the ways we can enable and disable signal processing and sets up a simple stereo amplifier from live input to live output.
D.2. SIGNAL PROCESSING IN PD

D.2.2 Simple signal manipulation

Here are a few simple things we can do with audio. Adding together two signals has the effect of summing (mixing) the audio. The addition occurs sample by sample as the stream is processed. Pd automatically sums signals that are connected to a single inlet.

As well as this we can add a scalar value to a signal which has the result of offsetting the signal by the specified value.

Multiplication is also performed sample by sample and a multiplication by a scalar value has the effect of scaling (amplifying) a signal. The multiplication of one signal by another results in a complex behaviour known as ring modulation.

D.2.3 Audio objects for generating signals

Here are some very useful signal generating objects. osc~ generates a pure sine tone. phasor~ generates a ramp waveform. This waveform has a rich frequency
content. \texttt{noise~} generates white noise and is very useful for experimenting with filters. \texttt{readsf~} is a sound file playback object. It is relatively simple to use but needs some utility objects to provide a file selection dialog box.

![Figure D.12: Basic Signal Generators](image)

### D.2.4 Adjusting the range of signals

As with the numerical data you often need to adjust the ranges of signals. To do this we can use a similar linear range mapping by scaling and offsetting using audio rate mathematical operators. As we are dealing with signals we use the signal versions \texttt{*~}, \texttt{+~} and \texttt{expr~}. Example D.13 uses two oscillators and some simple range adjustment to create an FM tone generator. One oscillator (the modulator) controls the frequency of the other (the carrier).

![Figure D.13: Basic frequency modulation synthesis](image)

### D.2.5 Other simple manipulations on signals

Pd has a large array of filters to choose from and these are all relatively simple to play around with. The following example makes use of the \texttt{vcf~} object but
there are a number of others (hip~, lop and bp~).

![Figure D.14: Example of the vcf~ filter](image)

We also have some very useful operators for creating multi-tap delay lines. Delay lines are one of the only places where Pd allows loops of signal processing objects. In the example we are making use of this ability to create a feedback stage to the delay line.

![Figure D.15: Variable delay (vd~) with feedback](image)

Although it is perfectly possible to create a full implementation of a reverb unit with the basic operators in Pd, the algorithm tends to be very complex and instead we can use a single external object to generate reverb very easily. Example 2.20 on page 47 uses the ready-made freeverb~ object.
D.3 Sampling

Sampled audio (sections of pre-recorded or algorithmically generated audio) can be handled in Pd with two distinct methods:

- Allocation of system RAM to store a fixed amount of audio that can be read from and written to extremely quickly (table/array based)
- Direct streaming of audio from hard disk (disk based using the \texttt{readsf}\textsuperscript{~} object)

D.3.1 Why do we need both types?

Buffers

- Buffers are important when random access within the audio is required.
- Buffers are always of fixed pre-allocated size, limited by the amount of system RAM on the computer (approx. 10Mb per stereo minute of audio \@ 44100Hz).
- Buffers can be used as lookup tables for oscillators and functions.

Streaming

- Streaming is not limited by the size of your RAM so can allow playback of very large files.
- Streaming relies on the speed of your hard disk or storage medium and is slower to access randomly.
- Streaming to disk allows you to record for as long as you have storage space and the amount of time does not require pre-allocation.

D.3.2 Buffer objects: table and array

- The most important object to understand is the table and array because this is the object that is required by all other RAM based playback and recording methods. We must always allocate a buffer with a certain name and specify an initial file, size and number of channels.
- Simple playback from a \texttt{buffer}\textsuperscript{~} is achieved with \texttt{tabread}\textsuperscript{~} or (for more complex pitch manipulation) \texttt{tabread4}\textsuperscript{~}.
Appendix E

Audio driver technology: ALSA, Jack etc.

On the Linux platform audio there are a large number of audio sub-systems. OSS (Open Sound System), ALSA (Advanced Linux Sound Architecture), Esound Sound Daemon (esd), PulseAudio (http://www.pulseaudio.org/), FFADO (freebob), PortAudio. For studio work the most important driver platforms are ALSA, FFADO and PortAudio because they support very low latency (minimal delay between physical audio arriving at the software) and a wide variety of professional audio cards.

In traditional analogue studios equipment was often connected together via a patch bay. This was simply a huge bank of jack sockets that were connected to the inputs and outputs of all of the equipment (microphone wall sockets, mixers, fx processors, dynamics processors, loudspeakers etc.). With the patch bay the studio engineer could use very short patch cables to connect sockets on the patch bay together.

When digital audio came about and more importantly, when the majority of traditional studio equipment could be replaced by software on a single machine, some of the equipment inter-connectivity afforded by the patch bay was lost. The Jack Audio Connection Kit project was started to allow inter-connectivity between applications and in the Linux studio it is an extremely powerful and important part of the tool-set:

JACK is a low-latency audio server, and was written for POSIX conformant operating systems such as GNU/Linux and Apple’s OS X. It can connect a number of different applications to an audio device, as well as allowing them to share audio between themselves. Its clients can run in their own processes (ie. as normal applications), or can they can run within the JACK server (ie. as a “plugin”). — Jack Project, http://jackaudio.org/

JACK is a server, sometimes called a daemon program. Once running it
APPENDIX E. AUDIO DRIVER TECHNOLOGY: ALSA, JACK ETC.

hides away in the background and performs its tasks without a great deal of interaction with the user. The audio applications that the user works with connect to the jack server and use it to interact with each other.

Due to other applications' dependency on the JACK server, JACK is usually the first audio application that you run. However, lots of software will automatically start JACK with the settings that you specify.

In order to prioritise performance of audio on the Linux platform a special modification to security settings in the kernel was made. This is called the Real-time kernel pre-emption patch. On most audio related distributions of Linux this has already been set up so it is usually nothing to worry about.

The JACK server can be started directly from the command line with the **jackd** command. It has a large number of parameters that specify precisely which audio card backend (ALSA, OSS etc.) will be used and many options for each individual backend. Entering **jackd --help** results in the following listing:

```
$ jackd --help
usage: jackd [ --realtime OR -R [ --realtime-priority OR -P priority ] ]
     [ name OR -n server-name ]
     [ --no-mlock OR -m ]
     [ --unlock OR -u ]
     [ --timeout OR -t client-timeout-in-msecs ]
     [ --port-max OR -p maximum-number-of-ports ]
     [ --debug-timer OR -D ]
     [ --verbose OR -v ]
     [ --clocksource OR -c [ cycle ] [ hpet ] [ system ] ]
     [ --silent OR -s ]
     [ --version OR -V ]
     [ --nozombies OR -Z ]
-d backend [ ... backend args ... ]

The backend can be 'alsa', 'coreaudio', 'dummy', 'freebob', 'oss' or 'portaudio'.

jackd -d backend --help to display options for each backend
```

To see the specific options for the back end you are using you specify **jackd -d backend name --help**. So for using the ALSA backend we specify **jackd -d alsa --help** and are presented with the following:

```
$ jackd -d alsa --help
Parameters for driver 'alsa' (all parameters are optional):
-C, --capture Provide capture ports. Optionally set device (default: none)
-P, --playback Provide playback ports. Optionally set device (default: none)
-d, --device ALSA device name (default: hw:0)
-r, --rate Sample rate (default: 48000)
-p, --period Frames per period (default: 1024)
-n, --periods Number of periods of playback latency (default: 2)
-H, --hwmeter Hardware metering, if available (default: false)
-M, --hwmonitor Hardware monitoring, if available (default: false)
-x, --xrunmode Soft-mode, no xrun handling (default: false)
-n, --nperiods Number of capture channels (default: 0)
-o, --outchannels Number of playback channels (default: 0)
-i, --inchannels Number of capture channels (default: 0)
-I, --input-latency Extra input latency (frames) (default: 0)
-O, --output-latency Extra output latency (frames) (default: 0)
-X, --midi ALSA MIDI driver (seq|raw) (default: none)
```

The list of options is very daunting but typically not many are needed because the default settings are sensible. A server is usually started with something similar to **jackd -R -d alsa** and results in something like the following:

```
$ jackd -R -d alsa
```
loading driver...
creating alsa driver... hw:0|hw:0|1024|2|48000|0|0|nomon|swmeter|--|32bit
control device hw:0
configuring for 48000Hz, period ≈ 1024 frames (21.3 ms), buffer ≈ 2 periods
ALSA: final selected sample format for capture: 32 bit little-endian
ALSA: use 2 periods for capture
ALSA: final selected sample format for playback: 32 bit little-endian
ALSA: use 2 periods for playback

The command line is useful for fully understanding JACK but typically you will be using a GUI based application to interact with it such as JACK Control (example E.1).

Figure E.1: JACK control to interface with the jackd server and make connections between software
Bibliography


Glossary

**Acousmatic** A listening stance favouring sound over sight. Normally this means the perception of sound without trying to attribute source or cause (a very impersonal listening, quite scientific in fact). However we are clearly going to bring our experience to bear upon any listening situation. The Acousmatic stance asks that we ‘listen in’ and focus on the sound itself (its energy, motion, colour).. 6

**acousmatic-potential** Normally the most audible characteristic of a sound that might best be modified. Acousmatic potential is how clearly a sound ‘gives away’ its characteristics. Consider a sustained 10 second sound that rises in pitch from C3 to C4 with some harmonic profile. Unless we want our piece to be going up and down like a roller-coaster, this sound is not going to feature in your work as ‘the theme’ but its acousmatic potential (sustained, harmonically pitched and rising) suggests a number of strategies. It is best to take these strategies separately.. 4, 12, 75, 80

**ASCII** American Standard Code for Information Interchange.. 96

**BEAST** The Birmingham ElectroAcoustic Sound Theatre is the performing wing of the postgraduate and staff composers at Birmingham University. Adrian Moore completed his PhD there under the tutelage of Professor Jonty Harrison.. 44

**bricolage** Construction drawn from a vast array of diverse objects. More commonly associated with the visual and sculptural arts. Levi-Strauss (1994) and Boulez (1990) have used the term bricolage to denigrate sonic art made from manipulations of recorded sounds.. 71

**DAW** Digital Audio Workstation.. 7, 17, 43, 48, 70, 73, 74

**eq** Equalisation is always associated with traditional filters such as lowpass, highpass, lowshelf, highshelf and notch. You can not bring out or dampen frequencies that are not there in the first place, but eq helps balance your sounds in the mix.. 43, 76, 83, 90

121
**feedback** Feedback is simply the re-routing of the output back into the input. Normally this happens after the output signal has been attenuated. Care should be taken when using feedback as it can lead to your system overloading. 48

**foldover** Often when a manipulation fails to do as you intend, frequencies go above 20Khz and ‘come back down again’. They are said to have been folded over back into the audible spectrum. 89

**frequency-domain** The frequency domain takes a sound and analyses it into small segments called windows. Each window is further analysed to find out exactly what frequencies, amplitudes and phases are present. Normally, graphs of the frequency domain are drawn Frequency (X) against Amplitude (Y) and do not show time. If time is to be shown, graphs are called sonograms and show Frequency (Y) against Time (X) with Amplitude shown as a heat intensity shading. 61, 64

**fricative** Vocal sounds made by the close vibration of vocal articulators. (Examples are ‘sss’, ‘zzz’, ‘thth’). 77

**Glitch** Glitches are precisely those sounds that would, under normal circumstances, appear unwanted. Often associated with low frequency amplifier hum and crackle or short noise bursts. 19

**GRMTools** A selection of plug-in modules for DAWs made by the Groupe de Recherches Musicales in Paris. A time-domain set was released in the early 90s followed by VST plugins in 1997, a spectral set working in the frequency domain early in 2000 and a release of three new tools in 2010. 39, 42, 63

**MIDI** Musical Instrument Digital Interface. A digital code that stored the essentials of a score: pitch, duration, timing alongside a few musical parameters such as aftertouch. This code enabled synthesizers to be linked together and eventually for keyboards to become an input device to a computer. 10

**OSC** Open Sound Control. Developed as but one of many ‘sons of MIDI’, Open Sound Control affords the communication of structured data between devices set up to send and receive information tagged in this way. Open Sound Control is common amongst many well known hardware and software devices. 100

**plosive** Vocal sounds made by first blocking the vocal tract so that air flow ceases. When air is released an explosive sound is produced. (Examples are ‘t’, ‘d’, ‘k’, ‘t’ ‘b’, ‘p’). 77
**post-fade** A tap on an audio channel after the fader, therefore affected dramatically by the state of the actual output fader. 68

**pre-fade** A tap into an audio channel on a DAW that diverts a quantity of sound before it reaches the fader routed to the mix bus. Very useful when you want specific control of wet and dry outside of the effect plug-in. 68, 69

**sound-object** Normally a combination of sounds that have a strong beginning, short sustain and timely release. 21

**stereo** A stereo recording is taken with two microphones. Sounds arriving at each microphone may vary in terms of intensity and time of arrival depending on the stereo recording technique used. It is these small differences which, when played back over two loudspeakers, replicate the spatial position of the sound. 6

**strata** Strata is a term we use to delineate spectral levels of sound. The most obvious strata is the canopied and rooted texture setting where a low bass drone and a high frequency sustained sound ‘encapsulate’ the spectral space. 39

**time-domain** The time domain is our traditional view of sound as a moving waveform graphed Amplitude (Y) against Time (X). 45, 61, 79
Glossary