Perspectives on Practice.

Chapter 35: Algorithmic Trajectories

Alex McLean, Roger T. Dean

We jointly designed and edited this volume because of our complementary, overlapping yet highly contrasting backgrounds (we have performed together and met first in the context of music research). The contrast between us stems both from our differing time frames of involvement, and from the fact that AM makes music primarily (usually solely) via a computer and in real-time whereas RTD is an acoustic instrumentalist (particularly keyboards, often with computers), and a composer (offline) as well as improviser (real-time). While AM was using computers from an early age, and began serious programming around 1986 (aged 11), RTD first used a (desktop) computer in around 1982 (already aged more than 30).

So in this final Perspective on Practice, we will discuss our own experiences and the development of our current enthusiasms. We hope that brief consideration of these trajectories will have some interest for readers seeking to engage with the breadth of our field of algorithmic music. We drafted our own sections, and then jointly edited the chapter, providing a brief conclusion; we also took advantage of helpful suggestions from external reviewers. See Note 1 to this chapter for information on cd and other sources of the music mentioned in the two authors' sections that follow.

Roger Dean (and my ensemble austraLYSIS):

My algorithmic music experiences centre on real-time use of computation in jazz and freer improvisation, but also contribute to my compositional work. If we consider that one thing that can distinguish an algorithm from a traditional sequencer is real-time mutability of timbre, pitch and event timing, then my first experiences of algorithmic music making were with the DX7 Yamaha FM Synthesiser, released in 1984. I still occasionally this pioneering instrument, but subsequently it was complemented in my

performing armoury by a succession of sampler instruments (EPS, Kurzweil) with parallel capacities. The DX7 was one of the first affordable digital instruments to permit real-time timbral and pitch manipulation, and its sequencer functions could be transformed while running. I found the timbral transformation very stimulating, and used it both within improvised solos in jazz (e.g. with Graham Collier Music on a 1985 album) and with my own European group LYSIS. My piece *Alela* (1986), was an algorithmic 11/8 rhythmic and harmonic accompaniment with which instrumentalists could improvise. The 11/8 pattern was transformed or rotated in ways akin to *Clapping Music* or *Piano Phase* of Steve Reich; and tempo and timbre (pitched vs unpitched) might change in performance. I found this broad potential for variability one of the most appealing aspects of algorithmic music; at the time I saw less point in using absolutely deterministic algorithms. This is not a surprising stance for an improviser. Indeed, between 1970 and 1980 I was (temporarily) opposed to the idea of making fully notated compositions.

In the 80s I experimented with some of the early live algorithms for improvisers, such as Laurie Spiegel's Music Mouse and the program M. But my usage became more serious after MAX was released, when soon after this I was made aware of it by my colleague the late keyboardist and researcher Jeff Pressing (see for example (Pressing 1988, 1994)). My usage was subsequently aided by collaborator Greg White, a member of austraLYSIS, into which I converted LYSIS on migration to Australia in 1988. MAX is now a well-known graphic object-oriented programming language for music, initially note-based , later (with MSP: MAX signal processing) adding sound-based music and image (with Jitter). It remains my core platform together with several others.

As implied already, I found that my distinct perspectives as instrumentalist, improviser and composer all found application in algorithmic music making. Much of what I have written as algorithms constitute performing interfaces or hyperinstruments, that I can use while also playing the piano, or that run fairly automatically with little intervention. I also write algorithms which require full time attention, and since 2014 I have also performed live coding with Gibber.

Algorithms in principle can be applied to any aspect of music. For me, an early appeal of algorithmic music was to make a range of 'rhythm machines' (see (Dean 2003)) which I used interactively

while playing the piano. Some operated on isochronic rhythmic units (making pulsed rhythms of various complexities), while others treated rhythm as a continuous variable (creating greater irregularity, sometimes essentially without patterns). In MAX, less than 20 objects are needed to produce metrical patterns, or repeating patterns of any diversity: each unit can be an integer multiple of an isochronic pulse, each can have a different duration, or some may repeat. The same patch can allow the pattern to be varied every time it repeats, when required maintaining an initial pattern of shorter-longer (sl) events, such as sslls even though every s and l duration might differ each time the pattern is enunciated. This work complemented my 1980s acoustic work on multiplicity of rhythms within an improvising group and on competitive metrical and pulse shifts, somewhat akin to Elliott Carter's metric modulation or the earlier complexity of Conlon Nancarrow. I also made a transformed (perhaps contorted) Drum 'n Bass generator algorithm, which is functional both as an autonomous automaton, and under user input (Dean 2003). Most recently Andy Milne and I have developed MeanTimes (now part of the freely available software Xronomorph), a MAX implementation of research ideas on the nature of well-formed rhythms that go well beyond isochronic pulse assemblies and metrical concepts of well-formedness, and may contain several disparate pulse rates and components, even to an extreme where there is no shared pulse amongst any of the metrical or hierarchical levels (Milne and Dean 2016).

Post-minimalism (spearheaded and coined by William Duckworth) is a compositional approach for which algorithms are ideal. It encourages slow melodic transformation, within and without tonality, alongside the other features. I used such algorithmic approaches to make *The Egg The Cart The Horse The Chicken* (2001) and *Mutase* (2008), which has also been used in empirical studies of music perception. In *MultiMutase* (2008) and *Ligating the Rhizome* (2006) I used chain annealing techniques to allow various simple forms of 'cross-over' between melodies.

I emphasise variability, but this does not preclude algorithmic approaches which from the perspective of music theory are highly constrained. For example, key features of Serial Composition using the techniques pioneered by the Second Viennese School (Schoenberg and his colleagues) can readily be coded, and I made such an algorithm (the Serial Collaborator (Dean 2014)), creating multi-handed piano music, which operates either autonomously or interactively. It can even be applied to tonal

motives. I also used related algorithms for pitch, dynamics and rhythms in some sections of my orchestral piece *SonoPetal* (1995; commissioned and premiered by the Australian Chamber Orchestra). Simple forms of machine listening contribute to many of my algorithmic patches (see also (Dean 2003).

My most recent work on algorithmic music has focused on using time series analysis models of note-based (and currently sound-based) music as generative devices. In my BANG (beat and note generator) algorithm (J. Creative Music Systems, Issue 2, in press 2017), time series models of an ongoing performance are regularly constructed (and cumulated). I have also embarked on deep learning neural net approaches to modelling music in performance, again for use in BANG. My currently developing TIMETRANS (timbre event transformer) will apply similar processes within continuous timbral flux. Music generation in BANG then uses the models to simulate the embodied process, and the output may be subject to other transformation before being sounded. BANG is somewhat like Pachet's Continuator (Pachet 2003, Pachet and Roy 2013), except that BANG is based on time series analysis or deep neural nets, rather than the Markov chain approach, and mounting evidence shows that these provide distinct representations of music, which may be complementary (Gingras et al. 2016). The purpose of BANG is to encourage more post-tonal melody, harmony, and free rhythm (continuous temporal durations) in the output, which will always diverge substantially from the input. BANG presently uses a multivariate approach with four music parameters: pitch, key velocity (a surrogate of loudness), note and chord duration, and event inter-onset interval. These are jointly modelled in the BANG algorithm, giving a model which simultaneously predicts all four outputs, and which allows for their mutual influences. Memory functions within BANG permit a wide range of hybridisation of different periods of the current (and earlier) performances, according to the choice of the instrumental co-performer. It may also operate autonomously.

Algorithmic manipulation of timbre and its spatialisation (see chapter by Schacher) is used in my solo MultiPiano performances, involving piano and computer producing multiple sonic strand: I use at least a 4.1 diffusion system whenever possible. Similarly, Jitter permits what I call algorithmic synaesthesia (Dean et al. 2006, Sagiv, Bailes, and Dean 2009), in which algorithmic processing and/or

input data are shared between sound and image. This is analogous to the impressive laser, sound and video works of Robin Fox and others.

Within austraLYSIS, my ongoing creative ensemble, several artists contribute major inputs (Dean and Smith, 2003). Figure 1 illustrates the opening of one of our intemedia works. Hazel Smith, our poet and performer collaborates on numerous works involving spoken, performed and/or displayed text, and since 2001 we have pursued algorithmic text manipulation within our Text Transformation Toolkit, developed in Python. Published works such as *Instabilities 2* are amongst the results. David Worrall collaborated in developing this Python toolkit, and is well known both for his pioneering work computer music in Australia, and in (algorithmic) sonification. Sonification is the sonic analogue of visualisation, processes for (hopefully accessible) representation of data content respectively in sound and image. With David and colleague Greg White I made a 22:4 spatialised algorithmic sonification of multichannel brain EEG recordings (2002, ICAD website). Greg also contributed technical guidance in the early stages of austraLYSIS' application of MAXMSP, Flash and Director (e.g. our *Walking the Faultlines*, 1996-7, published by ICMA) and *Wordstuffs*, a 1997 1Mb (!) floppy disk work occasioned by the Australian Film Commission and still available on the Australian Broadcasting Corporation (ABC) web site).

<Figure 1 near here>

White's recent thesis (White 2015) describes the trajectory of his work, often in our context. Alongside using Fibonacci series for pitch and rhythm in an austraLYSIS Electroband piece, Greg's notable early contributions (*Silence of Eyes*, *Glass Bead Game*, and *Scrolls of Time*, from 1995-2002) involved live performer interaction with the computational system and continuously generated score material which the performer then realised, be it text (*Silence of Eyes*) or musical. For *Glass Bead Game* the felicity or precision of fulfilment of the comprovisation score (i.e. improvising in ways which are consistent with the score) was assessed in real time, with consequent changes in the next score output. With *Scrolls of Time*, the challenge was to perform in real-time more exactly what the score progressively indicated.

Finally, I return to my own priorities, aside from those of austraLYSIS. In the temporal domain I distinguish influence and interaction from the performer to the algorithm, in the sense defined by Edmonds (Boden and Edmonds 2009): where influence is a delayed consequence of a performer action, while interaction involves an algorithmic response which is almost instantaneous, such that it may well be detectable by the performer, and perhaps by the audience. As developed also by others such as Laurence Casserley (Dean 2003), I do timbral manipulation by DSP with various time delays. I use a continuously updating 3 minute 'buffer' of recent sound. Live algorithms remain a primary performance vehicle for me, but I have also written JavaScript code exploiting the audio API for in-browser algorithmic sonic manipulation by users in installations and across the web (for example, in *Long Time No See*, a 2013 work by a team lead by Keith Armstrong).

Overall, I see that the early stages of my own algorithmic music, and I think also that of many others, were strongly driven by the appearance of new technologies at affordable prices. Ideally, one might spend the 10000 or so hours considered necessary for 'expert' performance in any field on aspects of algorithmic performance, in order to gain such 'professional' level facility. Since the advent of MAXMSP, the plethora of possibilities has been such that choices and enthusiasms can drive what I do; again I think this is a shared experience. I repeatedly engage a research-led-practice, practice-led-research cyclic web (Smith and Dean 2009). This is not to say that technologies, or artistic innovations by others, have no impact. It is to say that apart from computational speed, most of the challenges one would like to erect in making algorithmic music can in principle be overcome with sufficient coding effort. Algorithmic music has come of age in this respect, as in many others, which is one of the reasons I hope the present book is timely and will be useful to our colleagues and our listeners.

Alex McLean:

Many algorithmic music practitioners, including Roger as just described, have come to algorithms looking for ways to extend their music practice. For me it was the other way around, as a young computer programmer obsessed with the world of code, but yearning to connect more with the 'outside world', and finding music to be an ideal outlet. While working during the first internet boom in the '90s, programmers

were emerging from an anti-social stereotype, but still programming was quite far from being seen as creative in the public mind. Even discussion within the algorithmic art field was obsessed with questions around the authorship and autonomy of computers, with little heed paid to the creativity of the programmer. The difference was stark between this wider view of code as technical and mundane, and my own experience of it as an expressive, revelatory and humbling medium, its very use requiring rigorous self-reflection of ideas and intentions.

My first experience with algorithmic music was encouraged by software artist Adrian Ward. Ade had already produced generative artworks parodying mainstream software (which would later lead into the release of the award-winning Auto-Illustrator), and was starting to move from making Autechre-influenced techno in old-school tracker software, into generating music with code. We started writing software to make music together, and collaborating on events as VXSLAB, on 17th June 2000 advertising one with "Pure Perl code generating 'dance music' and ambience -- some of this Perl will be written live". Collective memory is unsure what actually happened there, but our early motivation clearly centred around bringing programming into the public realm. Adrian and I began performing as `Slub' later that year, joined by long-time collaborator Dave Griffiths in 2005, and we became known for projecting our screens, exposing our handmade interfaces.

As young, idealistic digital artists, Ade and I wrote "The Generative Manifesto", partly inspired by Ade's interest in the Fluxus movement. This was presented at the Institute for Contemporary Arts in London on the 23rd August 2000, with Ade shouting out the words, while I (being shy) ran one of our music-generating Perl scripts to underline each point:

> Attention to detail - that only hand made generative music can allow (code allows you to go deeper into creative structures)
>
> Realtime output and compositional control - we hate to wait (it is inconceivable to expect non-realtime systems to exhibit signs of life)
>
> Construct and explore new sonic environments with echoes from our own (art reflects human narrative, code reflects human activity)

Open process, open minds - we have nothing to hide (code is unambiguous, it can never hide behind obscurity. We seek to abolish obscurity in the arts)

Only use software applications written by ourselves - software dictates output, we dictate software (authorship cannot be granted to those who have not authored!)

Our emphasis on human authorship in algorithmic music, on celebrating creativity in computer programming, and on a live and open approach, is clear. Our aim to have people dance to our code was fulfilled late 2001, at the legendary Paradiso club in Amsterdam during the Sonic Acts festival. Ade and I controlled our generative music scripts from the back of the room, with VJ (video jockey) Pfadfinderei controlling visuals from the front. Being able to watch the crowd go crazy without knowing where the music was coming from, was a great moment, which has fed into my ongoing ambivalence with projecting our screens. With rare exceptions however, we always have, although often obscuring the code by projecting our screens on top of one another. Balancing the need to be open and forthright about our methods, while keeping focus on the resulting music, has always been difficult.

As a band, Slub developed outside academia, through regular performances at digital arts festivals, and two dearly missed venues in London; the KLF's Hoxton Foundry and Siraj Izhar's Public Life. Nonetheless significant crossover between the fringes of electronic music and academia lead me to submit papers to academic venues such as Music Without Walls in De Montfort University, 2001. Here I was amazed to discover that many academics in computer music looked upon dance music with derision, dismissing it as popular, and rejecting the notion of repetition. One exception was Nick Collins, presenting a seminal paper on breakbeat science (Collins, 2001). [Nick Collins and I later became co-founders of Chordpunch algorithmic music record label and the Algorave event promoters, hopefully helping improve the landscape of dance-oriented computer music.]

A major turning point came by invitation to the Changing Grammars symposium, organised by Julian Rohrhuber and Renate Wieser (both are authors in this book, as is Nick Collins who was also present). This symposium brought together people interested in what was variously called just-in-time programming, on-the-fly programming and live programming, and acknowledging that there are

important nuances in these terms, now generally referred to as live coding, with good coverage in this book (including by Roberts and Wakefield). At this moment Ade had made his MAP/MSG live coding system (a working parody of MAX/MSP), Ge Wang was releasing ChucK, and users of SuperCollider 3 were beginning to explore the possibilities of live interpretation of code in music making. Interactive programming has always been possible in computing, but there was certainly something in the air; with all this activity live coding was ready to come together into a community of practice. In a smoky club at the close of the event, the Temporary Organisation for the Promotion of Live Algorithm Programming was born, with a manifesto hastily drafted on the plane home (see Haworth's chapter for some revealing ethnographical work on this document). Many of those present have been through grad school, myself included, and are now increasingly senior academics, enjoying seeing where the next generation of young students are taking things.

Following our involvement with the founding of the TOPLAP live coding collective, Adrian and I decided to delete our old Slub system in 2004, continuing only with live coded performances from then on. I developed feedback.pl, a Perl live coding environment, so named because the live coded process was able to modify and re-interpret its own code. In practice this allowed the code to become a dynamic user interface much in the same way as a Max/MSP or PureData patcher language allows, writing code that updates data represented within the code as it is continually updated. The shift to live coding meant I was able to explore collaborations in Jazz improvisation, starting with percussionist Alex Garacotche, who pushed me towards the ideal of live coding from scratch, where live coding ideas are formed and implemented during a performance. This addressed the free improvisation ethic well, but was impractical when coding in Perl, it just took too long to start making sound, or make changes in response to those of an instrumental co-performer. When I finally succumbed to the call of academia and became a research student with Prof. Geraint Wiggins in 2006, this gave me the time and resources to explore functional programming and alternative models of computation that lead to the TidalCycles system described in my chapter with Thor Magnusson in the present volume (and also discussed by Wiggins and Forth in their chapter). Finally with TidalCycles I feel I am able to improvise with instrumentalists as equals, including through collaboration with percussionists Paul Hession and Matthew Yee-King.

I have not said a great deal about the actual algorithms I have explored in my music. This is in part left to the chapter on pattern in this volume, but I will say a few words here. I have not instigated an archaeology of my old code bases, which I fear may not be accessible on the stack of deteriorating hard drives in my studio, but can pick out some memories. I had an early interest in simple agent-based automata, such as flower.pl exhibited at Sonar festival, where agents travelled around the petals of an abstract flower, triggering sounds. Slight interaction between agents at junctions was enough for the agents to fall into a repeating pattern. Another obsession was with fork bombs, where processes split in two, rapidly taking over a host computer until it crashes; my performances often concluded with sonified fork bombs. In general, `glitch' was embraced in Slub performances, the hand-made nature of our software leaving it prone to audio drop-outs and synthesis artifacts, although because we knew the software intimately, only very rarely unintended crashes. Fredrik Olofsson appreciated this aesthetic, and even implemented a SuperCollider plugin, called Slub, to purposefully overload the host computer to induce glitching, still available in the official SuperCollider library of extensions. My main obsession throughout has been on different aspects of patterning such as polymetric sequences, reflective and rotational symmetries, and the endlessly fascinating interference patterns that emerge from combining patterns in different ways. Engaging with pure functional representations has taken this obsession to a new level in TidalCycles, and I am now happily surprised by what others produce with it almost on a daily basis.

In recent years, my collaborations (see Figure 2 for an example) have diversified and become increasingly important to the development of my practice (McLean). Work with choreographer Kate Sicchio, live artist Hester Reeve, performance artist Susanne Palzer, audio-visual noisemaker xname, and many one-off musical improvisations have allowed me to consider the bridge between my abstract algorithmic patterns, and to corporeal views of performance. A particular influence has been working with mathematician, philosopher and textile artist Ellen Harlizius-Klück, weaver and live coder Dave Griffiths and interlocutor Emma Cocker, looking at the correspondence between patterns in weaves, computation and music in our Weaving Codes, Coding Weaves project. A common thread in all this work

is looking for ways to ground computer programming in corporeal experience, through dance, movement or cloth.

<Figure 2 near here>

In summary, my motivations in algorithmic music have in large part been through a need to express myself and connect with people through a medium. Perhaps a greater motivation is the direct experience of algorithmic music, particularly of exploring and enjoying music beyond my imagination. Computer programming is often used to model external phenomena, but in my musical practice its usefulness is in-of-itself. When writing code to make music, I am able to describe an experience of music that I have not yet experienced, and cannot imagine until I hear it. For this reason, I feel the creativity of writing algorithmic music is just as much in listening it is in coding. Writing code feels like series of a syntactical twists and combinations, and the physical and cognitive result can only be known by experiencing the output. This is of course not unique to algorithmic music, indeed with reference to Paul Klee, Timothy Ingold (Ingold 2010) argues that all making is about *following* material, rather than hylomorphic imposition of ideas onto the world.

Outlook

Hidden under the surface of many of our comments, and emerging more strongly against this concept of hylomorphism (which has both philosophical and computational connotations) is the idea of recursion: a computational process generates a result which is immediately fed back into a repetition of that process, continuing until some predefined terminating condition is reached, a user intervenes, or a machine stops working. Recursion is also close to, or at least applicable to many of the ideas discussed in other chapters.

Perhaps algorithmic music at large can also be seen as many simultaneous large scale recursive processes, where outputs are not only used by their initiator, but by numerous other people. Internet exchange, web crawling or scraping, database exploration, and many other current computational processes facilitate this. In this image of algorithmic music it may be a long time before one result

returns, however transmuted, to the original creator and their code (no doubt already significantly changed itself); but this may happen.

Perhaps this is merely a microcosm of creative processes across the arts and sciences at large. Or perhaps it is something in which algorithmic steps are necessary to ensure that the eventual range of outcomes is as wide ranging as it can conceivably be. This book encourages hope that creative opportunities continue to expand and diversify. We certainly hope that the reader who has reached this point in our book will feel inclined to explore and contribute further; as we ourselves continue to do.

Note 1. Pointers to selected published recordings of music by the editors:

RTD: DX7 work with Graham Collier is on 'Something British' (1985; Mosaic LP GCM 871, re-issued on CD). austraLYSIS' Moving the Landscapes (Tall Poppies TP007, 1992); and six subsequent CDs on this label include algorithmic work, particularly MultiPiano (2012; TP 225), and History Goes Everywhere (2015). Algorithmic work with text performance (with Hazel Smith) and involving voice manipulation, is on two CDs on Rufus, and several intermedia works for radio, web and CD-rom. Noise sculpting is on the CD-rom with Dean's book Hyperimprovisation, as well as on several web works. Algorithmic image as well as sound manipulation is in numerous collaborative web pieces, installation works and an algorithmic digital novel (*novelling* : Binary Press, 2016), comprising text for reading, sound, and video. Besides Hazel Smith and austraLYSIS, key collaborators include Keith Armstrong, Will Luers and others. Links to the intermedia work, mostly freely accessible, and details of other releases, can be found at www.australysis.com.

AM: Many recordings of Slub live performances are freely available online via http://slub.org/, including one released via ChordPunch (CP0X08, 2011), and two via Fals.ch (#6374, #6444, both in 2002). My main published solo work is under the name Yaxu; the six track Peak Cut EP on Computer Club (DISK02, 2015), released as a limited edition USB stick containing a bootable operating system running the TidalCycles software and including the source code for the music. I also released a two track single

called Broken, again on Chordpunch (CP0X0D, 2013). The TidalCycles software that I initiated is available as free/open source software online (http://tidalcycles.org/).

References

Boden, Margaret A, and Ernest A Edmonds. 2009. "What is generative art?" *Digital Creativity* no. 20 (1-2):21-46.

Dean, R.T. 2003. *Hyperimprovisation: Computer Interactive Sound Improvisation; with CD-Rom*. Madison, WI: A-R Editions.

Dean, R.T., and H. Smith. 2003. "Sonic narratives: intermedia transformations in the work of austraLYSIS." *Australasian Music Research* no. 8:91-105.

Dean, R.T., and H. Smith. 2005. "The evolving technology of performance in the work of austraLYSIS, and the politics of co-operativity." *Sounds Australian* no. 65:16-21.

Dean, R.T., M. Whitelaw, H. Smith, and D. Worrall. 2006. "The Mirage of Algorithmic Synaesthesia: Some Compositional Mechanisms and Research Agendas in Computer Music and Sonification." *Contemporary Music Review* no. 25 (4):311-327.

Dean, Roger T. 2014. "The Serial Collaborator: A Meta-Pianist for Real-Time Tonal and Non-Tonal Music Generation." *Leonardo* no. 47 (3):260-261.

Gingras, Bruno, Marcus T Pearce, Meghan Goodchild, Roger T Dean, Geraint Wiggins, and Stephen McAdams. 2016. "Linking Melodic Expectation to Expressive Performance Timing and Perceived Musical Tension." *J Exp Psych:Human Perception and Performance* no. tba.

Ingold, Tim. 2010. "The textility of making." *Cambridge Journal of Economics* no. 34 (1):91-102.

McLean, Alex. Reflections on live coding collaboration. Paper read at In: xCoAx 2015: Proceedings of the Third Conferenc on Computation, Communication, Aesthetics and X. Universidade do Porto, Porto.

Milne, Andrew J, and R.T. Dean. 2016. "Computational Creation and Morphing of Multi-Level Rhythms by Control of Evenness." *Computer Music Journal* no. 40 (1): 35-53.

Pressing, J. 1988. "Improvisation : methods and models." In *Generative Processes in Music. The Psychology of Performance, Improvisation and Composition.*, edited by J. Sloboda, 298-345. Oxford: Clarendon Press.

Pressing, J. 1994. *Compositions for Improvisers : An Australian Perspective*. Melbourne: La Trobe University Press.

Sagiv, N., F. Bailes, and R.T. Dean. 2009. "Algorithmic synaesthesia." In *The Oxford Handbook of Computer Music*, edited by R.T. Dean, 294-311. New York, USA: Oxford University Press.

Smith, H., and R.T. Dean. 2009. "Practice-led research, research-led practice: towards the iterative cyclic web." In *Practice-led research, research-led practice in the creative arts*, edited by H. Smith and R.T. Dean, 1-38. Edinburgh: Edinburgh University Press.

White, G. 2015. *Towards maximal convergence*, Newcastle, Australia.