The Textural X

Alex McLean
a.mclean@leeds.ac.uk
Interdisciplinary Centre for Scientific Research in Music, University of Leeds, UK

Keywords: Computer Programming, Live Coding, Knitting.

Abstract: This paper considers the binding of analogue and digital forms in the context of computer programming. An argument is constructed based upon a knitting metaphor, relating patterning of wool with the functions of code over time. The relation between linear and cyclic time is considered, from the standpoint of the experience of programming, in particular the live coding of dance music. By way of illustration, example code demonstrating the weaving of analogue (continuous) and digital (discrete) pattern is shown, using pure functional code with visual examples.

xCoAx2013
Computation Communication Aesthetics and X. Bergamo, Italy. xcoax.org

1. Introduction

Somewhere between the Analogue and the Digital lies the whole experience of *texture*. Computers offer a cruel enactment of a wholly digital realm: a discrete, mathematical world stripped of gesture and emotion. Strangely, many who associate themselves with the term *digital art* approach the digital nature of their medium with disgust[1], and have constructed an analogue substrate, where code is hidden from view, and pixels are merged into beautifully anti-aliased, continuous shape. But even this substrate must stand the charge of being impoverished; human-computer gestures are increasingly reduced to prods and smears on flat glass. This charge extends to interactive digital art; see Figure 1 for an extreme example of the role spectators are reduced to in art and design exhibitions. It seems that in order to escape the digital environment, we have created a farce of the world outside[2].

1. For example, see Simon Penny's commentary on his artwork *Fugitive II* (accessed 20th January 2013).

2. This polemic should not get in the way of celebrating the many fine examples of interactive artworks which live between these extremes.

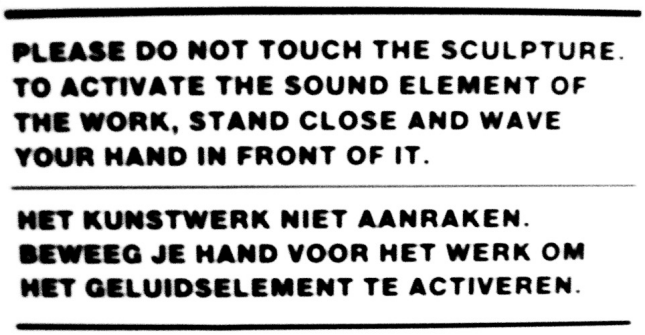


Fig. 1: Instructions for interacting with an artwork, seen at STRP festival, 2009.

Perhaps instead of focusing on either digital or analogue aspects, we should instead focus on how they mutually support one another in perception (Paivio 1990). After all, the shift from digital to analogue is a reversal of history; the discrete form of text emerged from imagery, and is made from imagery. Rather than thinking of computers as devices either for textual communication, or for systems that support gestural tools, perhaps we should use them to search for an illusive X which binds the two. As this X has continued to flow between us over evolutionary timescales, the vocal tract has developed as its primary conduit. Over this time our mouths have articulated not only to eat and breathe, but to form grunts, drones, chants and words; an organ for *digital* phonetic symbols, slurred into diphones and with *analogue* prosodic gesture and rhythm. We do not fully understand the operation of the vocal tract, but the trace of X is clear, in the poetic whole emerging from the simultaneously discrete and analogue articulation, intertwined in mutual support.

2. The loop

Along with the need to communicate with the voice, comes also the need to keep warm. Somewhat neatly, *knitting* provides metaphorical patterns and knots with which we may bind language with form. Knitting patterns are a kind of natural, domain-embedded programming language (Gold 2011), and mechanical computers and textile looms share early history (Babbage and Lovelace; Essinger 2004). In the following then, we use

knitting as a metaphor on which we build an alternate viewpoint of the experience of programming, with focus on time.

Time itself is an arrow, and we are propelled forward with it. It is also a circle, for example the cycle of life, returning to where it began. These two views are hardly reconcilable, in linear terms it is the future which comes to meet us, and in cyclic terms it is the past. So it is with knitting socks; a line of wool, five straight needles, a cyclic pattern tying loops into circles, the heel turns but eventually the sock emerges.

We often understand computers in terms of an algorithm (pattern), converting analogue inputs into digital discontinuities (wool into knots) and the form of text(ile) that results. But how often do we attend to the experimental possibilities of the loop? Nowadays computer processes rarely run to a conclusion, but loop continuously, oscillating in sympathy with human interaction. Perhaps we should consider software not as tools, applications, or frameworks for producing something, but as a fabric which captures the oscillations of hardware, to be experienced in its own right.

Notionally, the present moment is a durationless point in time. Experientially, a perceived moment has a duration of sorts, for example we do not experience sound in terms of states of airpressure, but in terms of fluctuations which deliver the experience of a discrete, momentary sound. We can extend this argument to the cyclic pattern of a rhythm, which may last from seconds to minutes. If the cyclic period of a rhythm matches with the duration of the present, we freeze, lost inside feedback. In this state, we step out of time, but also bring time into sharp focus; small changes amplified against a stationery ground. In terms of the sock, a repeated pattern forms vertical striations of purl against knit; a small change in the cyclic pattern causes a sharp discontinuity, making those striations appear smooth.

3. Knitting with time: Live coding

Live coding is the use of programming languages in exploratory work, where code is dynamically interpreted so that edits take effect without restarts. Live coders often work before a live audience, such as in improvised music performance (see figures 2 and 3, and also Collins et al. 2003). This is a radical departure from conventional software development, breaking down artificial barriers between technologists and creative users, and has taken electronic music research by surprise (Emmerson 2007). Time will tell whether the Live Coding movement will really contribute to fundamental widespread change, but it is useful to criticise technical practice (after Agre 1997), and look for points where the ongoing narrative of technological determinism may be broken.

Fig. 2: Dave Griffiths and Alex McLean live coding as two thirds of the band *Slub* (http://slub.org/) in Mexico City, November 2012. Their screens are projected behind them, so the audience can see their code, in-line with the TOPLAP manifesto (Ward et al. 2004).

Fig. 3: Audience dancing to the live coding performance shown in the previous figure.

In 1987, Nintendo trialled a knitting machine controlled by a computer games console. Industry commentators recall this as a hilarious aberration, a prototype quickly dropped after a bemused executive failed to find words to sell it to management. In this moment, knitting and computation, so close in Babbage and Lovelace's time, had the possibility of being reunited once more in console gaming. We can imagine this as a 1980's paradigm shift that never was, an ungovernable flow of scarves emerging from every child's bedroom. This could have created a very different expectation for human-computer interfaces, with progress towards interactions more textural rather than those prods and smears currently in vogue.

The knitting metaphor may still serve us well. Live coding music is very much like knitting with time. Time is a livecoder's wool, not so much in the sense of recorded tape, but more in terms of the line of a monster curve. The line is twisted, knotted and transformed by patterning structures, thereby creating new dimensions of experience.

Although knitting of socks is enjoyable, the real purpose of socks is to be worn. We wear code by running it, constructing environments that we listen and dance to. In dancing the encoded meter, we set the ground: we find an implied pulse and feel it with the whole body, where the pattern is experienced in contrast. By stepping into the music, we become part of the program interpretation. But, when we are finished, we are left with nothing. Live coders knit a live fabric, not an end product; we can touch it, but then it is gone.

4. Knitting with code

This line of thinking may be set more concretely in the practice of computer programming, by considering sourcecode as a pattern for physical experience. In particular, using *Tidal*, a domain specific language for musical pattern, embedded in the pure functional programming language Haskell. Pure functional programming is a familiar topic in computer science, and occasionally found in mainstream programming practice. What makes a programming language *pure* is that a function has no effect beyond turning one value into another value. What makes it *functional* is that values can be higher order constructions, such as "add 5" or "make twice as fast".

Tidal represents music as a pure function, which takes time as input, and outputs sound events. This maps from the single dimension of time into multidimensional dance music, and has a direct analogue with knitting thread into two dimensional texture. Both involve repetitive, looping patterns, forming a shape that fits the body.

In Tidal, the knitting of time into music is represented using the following datatype:

```
data Pattern a = Pattern (Arc -> [Event a])
```

In other words, a Pattern is a function from an Arc of time, to a list of events of type a, where a can be replaced with any other type. The above datatype makes use of the following type synonyms:

```
type Time = Rational
type Arc = (Time, Time)
type Event a = (Arc, a)
```

Time here is a represented as a rational number, of arbitrary precision. An Arc is a pair of Time values, to represent a start and stop Time. An Event is a value that occurs over a particular Arc.

A Pattern may behave in two distinct ways, depending on whether it represents discrete or continuous patterning. Figure 4 illustrates the behaviour of a discrete colour pattern in visual terms. A Pattern should only return events active for some part of a given query, although they may start or end beyond the query. Note also that the returned events may overlap, in order to represent polyphony.

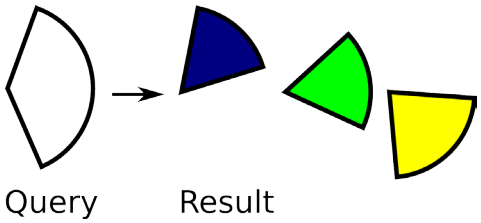


Fig. 4: A discrete colour pattern, showing that a pattern, as a function, may return a number of colour events active within the given arc, each occurring within their own arc.

This allows a simple representation of transitory values, each of which exists for a discrete period within a timeline. The timeline is notionally infinite, and we can probe for events using any Arc within it. As implied by the name *Arc* though, time is not only conceived as linear, but also cyclic. As Figure 5 illustrates, a cycle has a period of 1, which can be subdivided with arbitrary precision. This does not preclude polyrhythmic structures, but a fundamental loop, of period 1, is the focus. This accords with experimental evidence provided by London (2004), supporting his hypothesis that humans only attend to one meter at a time (although may have control over which they attend to).

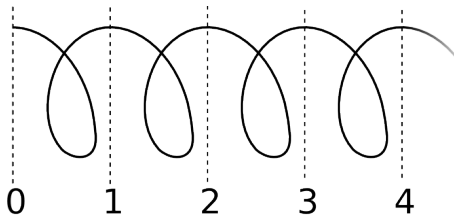


Fig. 5: A visual conception of a timeline as a spiral or coil, along which repeating patterns unfold and develop.

So far we have talked only of discrete patterns, but the same representation can be used for representing analogue, continuously-varying patterns. This relies upon the simple intuition that the closer you look at a continuous pattern, the more detail you are able to see. So, to represent a sinewave, a Pattern may return the average value of the given arc. In this way we are able to represent continuously varying values (as in Functional Reactive Programming; Elliott 2009) accurately, choosing what granularity or rate we use to sample values from it later.

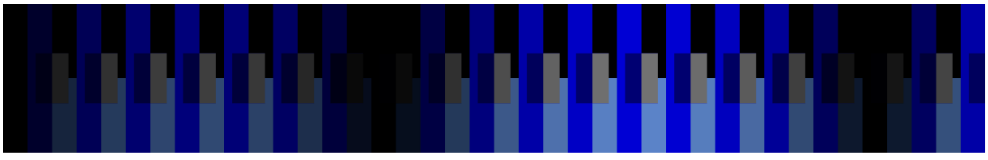
The distinction between these two kinds of behaviour is the same as the distinction between analogue and discrete views of texture, as discussed earlier. They are distinct, but can be combined in mutual support. Tidal is built around ways of using discrete and continuous together in rich, multidimensional, musical patterns. This amounts to the melding of the analogue and digital in computer language, but we will not go into further technical detail, instead turning towards some examples of use.

5. Tidal in action

It is difficult to get music across on paper, so in sympathy with the present medium, the following patterns will be of colour. Please consider the the horizontal axis as time, and the colour onsets and blends to construct temporal structures, which as music would be explorable through bodily movement.

The following code, shown above its output, multiplies a sinewave with a triangular wave (which has half the period), and applies the resulting signal to darken a sequence. The sequence here is described as superimposed sequences of colour, which are separated by commas. The important thing to observe is that simple continuous and discrete patterns can be combined, and that we can simultaneously perceive a continuous transition over a discrete pattern.

```
density 10 $ flip darken
             <$> "[black blue, grey ~ navy, cornflowerblue blue]*2"
             <*> (slow 5 $ (*) <$> sinewave1 <*> (slow 2 triwave1))
```

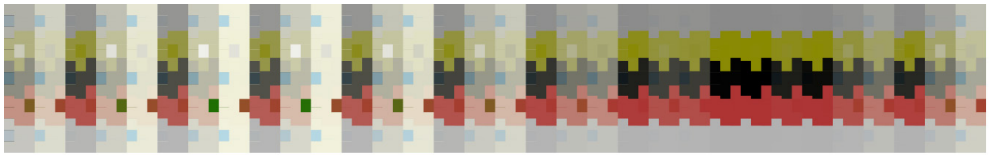


The following pattern is similar, but takes two sequences, and uses a sine wave to blend between them.

```
density 12 $ (blend'
              <$> "blue navy"
              <*> "orange [red, orange, purple]"
              <*> (slow 6 $ sinewave1)
             )
```

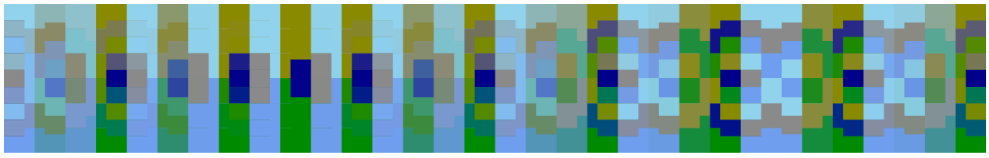
The following pattern uses a continous pattern to modulate the opacity of one sequence that has been placed over another.

```
density 32 $ flip over
             <$> (“[grey olive, black ~ brown, darkgrey]”)
             <*> (withOpacity
                  <$> “[beige, lightblue white darkgreen, beige]”
                  <*> ((*)
                       <$> (slow 8 $ slow 4 sinewave1)
                       <*> (slow 3 $ sinewave1)))
```



Finally, the following pattern blends between two instances of the same pattern, at different densities.

```
density 2 $
  do let x = “[skyblue olive, grey ~ navy, cornflowerblue green]”
     coloura <- density 8 x
     colourb <- density 4 x
     slide <- slow 2 sinewave1
     return $ blend slide coloura colourb
```



6. Conclusion

We have seen various family resemblances between knitting and programming, providing fertile metaphorical ground to explore the X of programming, taking an alternative view from the usual metaphors which are often born from commercial and military

contexts. In particular, it allows us to consider both the experience and purposes of programming in terms of binding analogue as well as discrete forms. The code examples above may be simplistic, but are offered in support of this view, where the composed, discrete text of source code may evoke rich experience, much as the text in a novel evokes rich scenes within a narrative. Where we work with such code live, to which groups of people come together to dance, we have a good place to search for the elusive X.

Bibliography

Agre, Philip E. "Toward a Critical Technical Practice: Lessons Learned in Trying to Reform AI." In *Social Science, Technical Systems, and Cooperative Work: Beyond the Great Divide* (Computers, Cognition and Work Series), ed. Geoffrey Bowker, Susan L. Star, Les Gasser, and William Turner. Psychology Press. 1997.

Collins, Nick, Alex McLean, Julian Rohrhuber, and Adrian Ward. "Live coding in laptop performance." *Organised Sound* 8: 321–330. 2003.

Elliott, Conal. "Push-pull functional reactive programming." In *Proceedings of 2nd ACM SIGPLAN symposium on Haskell 2009*. 2009.

Emmerson, Simon. "Postscript: the Unexpected is always upon us — Live Coding." In *Living Electronic Music*, 115. Ashgate Pub Co. 2007.

Essinger, James. *Jacquard's Web: How a Hand-Loom Led to the Birth of the Information Age*. Oxford University Press, USA. 2004.

Gold, N. "Knitting Music and Programming: Reflections on the Frontiers of Source Code Analysis." In *Source Code Analysis and Manipulation (SCAM), 2011 11th IEEE International Working Conference on*, 10–14. IEEE. 2011.

London, Justin. *Hearing in Time: Psychological Aspects of Musical Meter*. Oxford University Press, USA. 2004.

Paivio, Allan. *Mental Representations: A Dual Coding Approach* (Oxford Psychology Series). Oxford University Press, USA. 1990.

Ward, Adrian, Julian Rohrhuber, Fredrik Olofsson, Alex McLean, Dave Griffiths, Nick Collins, and Amy Alexander. "Live Algorithm Programming and a Temporary Organisation for its Promotion." In *read_me — Software Art and Cultures*, ed. Olga Goriunova and Alexei Shulgin. 2004.