# Surface as Structure: The multi-touch controller as map of musical state space

**Oliver Bown**
Design Lab,
University of Sydney,
NSW, 2006, Australia
oliver.bown@sydney.edu.au

**Daniel Jones**
Department of Computing,
Goldsmiths,
London, SE14 6NW, UK
daniel@jones.org.uk

**Sam Britton**
School of Arts,
Brunel University,
Middlesex, UB8 3PH, UK
sam@icarus.nu

## ABSTRACT

In this paper we present a new general approach to the use of multi-touch screens as musical controllers. In our approach the surface acts as a large hierarchically structured state-space map through which a musician can navigate a path. We discuss our motivations for this approach, which include the possibility of representing large amounts of musical data such as an entire live set in a common visually mnemonic space rather like a map, and the potential for a rich dynamic and non-symbolic approach to live algorithm generation. We describe our initial implementation of the system and present some initial examples of its use in musical contexts.

## 1. INTRODUCTION

In this paper we present a new general approach to the use of multi-touch screens as musical controllers. In our approach the surface acts as a large state-space map through which a musician can navigate a path. Our motivation is to find a way for the multi-touch controller to act as an intuitive and memorable representation of some underlying musical structure, rather than as a set of controls applied to elements within a musical structure which is defined elsewhere. In this way we hope to extend the potential of the tablet as a form of musical instrument, exploiting its novel properties. In particular we believe that through a zoomable map metaphor, we have a method by which a performer can have rapid access to huge amounts of data, such as an entire live set or even their entire repertoire, in a single surface. Our work so far provides an elementary sketch of these ideas, supporting our intuition that the approach is musically valuable. In this paper we present a set of design considerations surrounding the use of multi-touch surfaces (Section 2). We then describe our system design and implementation (Section 3). We then present a number of applications (Section 4) which are in progress, but sufficiently developed to suggest novel musical uses.



## 2. DESIGN CONSIDERATIONS

Commercial multi-touch devices have led to a proliferation of novel music performance and production systems. The majority of these systems model traditional electronic musical controllers such as banks of knobs and sliders, instruments such as a piano keyboard or drum pads and elements from Digital Audio Workstations (DAWs) such as waveform editors and loop-based sequencers. One of the most popular systems is TouchOSC [1], which allows users to lay-out standard knob, button and slider widgets, bound to custom OSC messages. TouchOSC's combination of simple elementary GUI elements with customisability for specific performance contexts has proven successful, and the distinction between controller and controlled system is clear (the user can program two-way binding between controller and controllable elements where necessary). Although largely similar, the iOS version of the original Lemur controller [2] distinguishes itself by providing a host of generative behaviours built into its control widgets, demonstrating how the tablet can further exploit its built-in processing capacity, acting in part as generative music system.

Several novel alternatives applications of the multi-touch surface context exist. For example a popular approach, which clearly diverges from the use of traditional instruments and controllers as reference points, uses the multi-touch surface as a way to define musical interactions between elements to produce generative effects. A typical approach following this format is to use properties relating objects in the Euclidean space of the surface – distance, angle, etc. – to musical properties – volume, pitch, tempo, etc. Although it has yet to be deployed for a tablet, the Nodal software developed my McCormack *et al.* [1] is a good example of the compositional power of such approaches.

Certain types of control are additionally afforded by the multi-touch format. Two-dimensional sliders provide parametric data control. The topological nature of 2D space is such that paths through a parameter space can be drawn that are circular rather than palindromic, a limitation of 1D space. This potential is well demonstrated in instrument layouts that use Euler pitch lattices (for example, [2]). These layouts allow performers to trace non-palindromic paths through a set of harmonically related pitches, which provides a musically meaningful structuring of control in a

[1] http://hexler.net/software/touchosc.
[2] http://liine.net/en/products/lemur/.

2D space.

The multi-touch context also has an inherently obvious coherence with polyphonic music performance, formed by assigning each finger to a musical voice.

In this paper we focus on the broader potential of these inherent features of 2D multi-touch control in defining relationships between surfaces and musical structures. We begin with the starting point of the multi-touch surface as a directly playable polyphonic musical instrument: each finger triggers a sound on touch-down, and sustains it until release, with potential changes to the sound while it is playing being caused by drag actions. (As we will show our approach can also be extended beyond direct sound generation, and could even be used, in the extreme, as a system for visual computer programming).

Rather than thinking in terms of fixed designs for instrumental surfaces, however, our interest is in methods for designing or automatically generating surfaces that in one way or another embody some aspect of the musical structure being controlled, either being derived from a given musical structure, creatively generated in order to produce a musical structure, or created in combination with some musical structure.

By musical structure we mean any arrangement of musical elements that guides the creation of an actual musical output. A score or actual musical recording is a particularly final structure, in the case of a score and less so in the case of an audio recording you may argue that there is still plenty to be finalised. Rule-based scores offer more flexible structures.

All instruments embody some aspects of musical structure. The placement of black and white keys on the piano reflects diatonic form. A guitar's strings are arranged to facilitate the performance of chords. Other instruments such as the auto-harp embody musically motivated design decisions, constraining musical choices to a limited set of harmonic relations. Whilst the latter may be more of a counter-example, most instruments successfully balance the freedom of open-endedness with these constraints, which can be thought of in the language of perceived affordances [3], the interactive offerings an object contains within it.

Computation provides other structuring methods: Markov models, for example, are statistical models that can be used to determine musical sequences based on the probabilities of transitions from one note or sequence of notes to the next, and are are a common strategy for music AI.

As district from traditional instrument interfaces our emphasis is on (a) the adaptiveness of the interface to different musical needs, (b) the capacity for the musical creator to be creatively engaged in the layout of the surface, and (c) using the 2D graphical surface to the best of its potential.

Our research question, then, is how can surface structures – the organisation of shapes in a 2D space – be used to represent musical structures in usable and creatively effective ways?

A number of design considerations were considered in arriving at the concept presented in this paper:

*Non-widget based* – It has been suggested that non-widget based interfaces may provide more natural performance contexts that those using standard GUI widgets (e.g., [4]). Further still, touch interfaces that require little or no visual feedback are seen by some as preferable for musical performance; the great majority of traditional musical instruments fit into this category.

*Singular* – Electronic performance interfaces can be divided between those that essentially employ a singular paradigm, such as the Korg Kaoss Pad [3] , and those that combine multiple GUI elements, such as TouchOSC. Although it would be pointless to argue that TouchOSC is unintuitive, it may be worth considering the utility of singular paradigm approaches.

*Hierarchical* – Hierarchical structural analysis can be applied to patterns in both shape and music and is arguably essential to our cognitive capacity in both domains. Therefore hierarchical structuring may be a useful way to think about developing a correspondence between the two.

*Visually Mnemonic* – Shapes are easily memorable and can be used as visual mnemonics for other entities such as sounds or other musical structures.

*Massive* – Graphical surfaces can contain visual structures of unlimited size, accessed through zooming and tracking. They can therefore provide controller surfaces with multiple elements combined in a single canvas most of which is hidden at any point in time.

*A Novel Niche for Generativity* – Touch interfaces that can be understood as topologically connected spaces are environments that can be interacted with using ones's finger but also using automated agents that are able to autonomously explore these spaces. In this way they have the obvious perceived affordance of being touchable, but also a design affordance according to which they can be automatically navigated by programmed agents.

## 3. SYSTEM DESIGN AND IMPLEMENTATION

The following description of a system design and implementation is based on a first attempt to capture the idea of extendable, visually mnemonic shapes as structured music performance interfaces.

*Surfaces* – We begin by defining a *surface* as a set of polygonal regions with non-intersecting edges (no edge of any polygon crosses any other edge of that polygon or of any other polygon, although edges may lie on the same line). Surfaces can be stored as sets of region definitions in JSON [4] objects for cross-platform use. In our experiments, we have generated surfaces using different techniques and rendered them on iOS and Android platforms either via OpenGL or native graphics methods. Tablet implementations include the ability to track, rotate and infinitely zoom surfaces, using direct two-finger transforms, with a toggle to switch between 'zoom' mode and 'playback' mode. The effect is similar to using an interactive map program such as Google Maps [5] .

*Voices* – A touch in any region creates a new voice and binds the voice to the currently touched region. Drag actions that traverse region boundaries cause the current re-

[3] See http://www.korg.com.

[4] JavaScript Object Notation. See http://www.json.org/.

[5] http://maps.google.com.

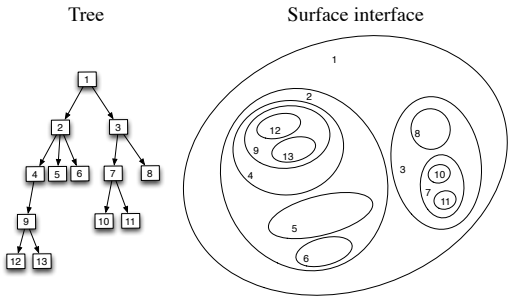


**Figure 1**. Relationship between the arrangement of shapes on a surface (right) and the tree representing the shape hierarchy (left).

gion associated with the voice to be updated to the new region. Each voice is monophonic, polyphony is achieved by multiple voices (i.e., multiple touch events). A voice has exactly one region set as its current region at any one time (any point in the space is bound to exactly one region). We say that the voice is *activating* that region. Multiple voices can activate the same region at the same time. Voices are assigned unique IDs and there is no limit to the number of voices that can be active at any one time (noting that automated playback agents can also act as voices, as well as human fingers).

*Messages* – All voice activation data is sent as an Open Sound Control (OSC [6] ) message to a receiving music computer. The message contains the type of voice, either "finger" or a string identifying a type of automated playback agent (e.g., "looper", "random walker"), the voice ID, a string representing the currently active region, which we will explain below, and optionally additional information such as how close the finger is to the centre of the region, touch pressure, etc. Except for this last type of information, which was not considered in the studies presented in this paper, messages need only be sent when a voice activates a new region or is released entirely. OSC messages therefore describe enter and exit actions on given regions.

*Hierarchical Structure* – Polygonal surfaces as defined above can be said to have hierarchical structure in the nesting of polygonal regions within other polygonal regions. In a simple polygonal picture of a face, for example, eyes and nose are nested inside head, and pupils are nested inside eyes. The spatial hierarchy can be represented as a tree, with head at the root, with two eyes and one nose descending from it, and a single pupil descending from each of the eyes. Figure 1 shows a simple example of the relationship between a tree structure and a surface.

As a voice traverses a surface, therefore, by entering, moving and releasing, its trajectory can also be described in terms of the traversal of the shape hierarchy (Figure 2). Our view is that such spatial hierarchies provide a sensible structuring paradigm bridging, on the one hand, the arrangement of shapes in a surface, and on the other, aspects of musical structure.

[6] See http://opensoundcontrol.org/introduction-osc

In the simplest case, hierarchical structure is not considered and the system simply takes the form of a spatial distribution of buttons, possibly resembling a piano keyboard or drum pad. In this case the potential for shaping the surface in usefully memorable or performable ways is of interest, offering the possibility of laying out musical objects for a performance in ways that embody musical sequences. But as well as looking at appropriate ways to layout musical relationships in 2D we are interested in ways to layout hierarchical musical relationships using 2D hierarchical spatial relationships, with the aim of making complex forms of control more intuitive, and exploring the potential of unlimited zoomability in shape hierarchies, and what that could mean for musical control at different scales.

A number of possible hierarchical structures in music make potential candidates for representation using surfaces, such as: structures deriving from grammatical analysis of music (as in [5]); levels of control in hierarchically nested signal chains, such as in groupings of devices in Ableton Live [7] ; sequences of operations on objects that can be written as reversible program instructions, for example, a transform applied to a sound, such as modulation in the context of synthesis.

### 3.1 Surface creation and structural mapping

In this section, the following surface design methods are considered. In the first case, Voronoi tessellation is used to derive a set of regions from a set of points. The relationship between points is determined in two different ways, either by a designer, who draws the points (at the same time as specifying their relationship), or by a process of hierarchical clustering, with a modified pruning stage to make appropriate trees. In the second case, shapes are derived from other sources and their hierarchical structures are extracted. In the third case, the musical data is used to derive shapes. Only the first method has been implemented in this paper.

These methods cover three general approaches to making surfaces: (i) procedurally – a generative process generates surfaces that have interesting abstract properties but no inherent associations with a musical system; (ii) designed – a user can in some way design a surface. Although the surface has no inherent associations with musical elements the design process may make those associations, for example the designer may derive a surface from analysis of a photograph of an orchestra; (iii) musically determined – the surface structure is derived from information about the computer music elements that will be controlled by the surface.

#### *3.1.1 Procedural shape generation: noise combined with hierarchical clustering*

Our initial experiments looked at procedural approaches to creating surfaces. Whilst procedurally generated structures may seem overly abstract for the context, our interest was whether structure generation procedures may be found that result in surface properties that could be of creative musical interest. The initial arbitrariness of mappings from such

[7] See http://www.ableton.com.

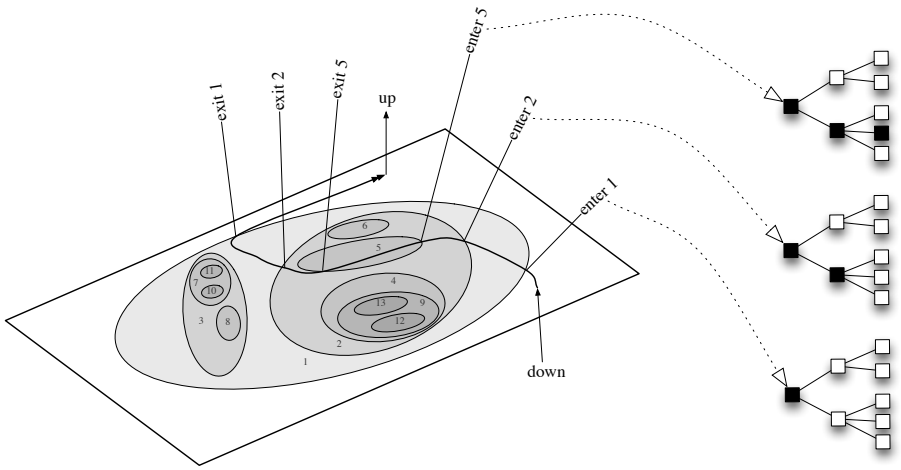


**Figure 2**. Example of a trajectory through a surface caused by a touch event, indicating the messages sent and the resulting tree activation states.

abstractly generated surface structures to musical control may not be a problem, since the performer's internalisation of this relationship is already an adaptive process.

A simple approach to procedural shape generation is to generate a set of points from which to derive both regions and their hierarchical relations.

Voronoi tessellation transforms any set of points into a set of convex polygonal regions such that any point inside a given region is closest to the centre point associated with that region. Through Voronoi tessellation, therefore, a set of points can act as a description of a surface.

Hierarchical clustering is a clustering technique that creates a tree of proximity relations from a given set of data points, rather than the specific categorisation into clusters resulting from most clustering algorithms. The tree can then be used to derive clusters. The hierarchical clustering algorithm begins by binding nearby points together to form subtrees and then proceeds by incrementally binding subtrees based on their cluster distance. In the result, each node has at most two children (i.e., these are binary trees), meaning resulting trees can turn out to be very deep. This can be corrected using customised pruning.

The combination of Voronoi tessellation and hierarchical clustering mean that both surface and tree structure can be derived from any given set of points.

Point distributions were generated using Perlin noise, which is used in the generation of natural landscape effects, and resulting region tessellations and trees were generated. Figure 3 shows one such generated surface. A colouring algorithm based on the position of nodes in the tree is used to indicate the tree structure. In this case, unlike the drawn example in Figure 1 the hierarchical structure does not clearly visually correspond to the nesting of polygons. Here the polygons can be thought of as nested but with shared edges between children and their parents.

### *3.1.2 Dynamic cellular shape design*

As an alternative to using procedural methods to produce point distributions, we constructed a simple drawing tool with which a surface designer could arrange points. The Voronoi tessellation was still used to derive regions from points, but in this case the designer could also manually specify the relations between points. The drawing tool was designed to be dynamic, so that points could be sprayed on a surface but would partly self-organise according to the relations specified by the designer, the idea being that they would relax into more or less regularly distributed structures, with the degree of relaxation being a parameter that could be chosen by the designer. This was achieved using a mass-spring model applied to points, with points connected by springs based on their relationship in the tree structure. This provided an intuitive and manipulable region-drawing tool, albeit only a rough prototype, which could be of additional interest if also allowed to run and modify itself during musical playback.

### *3.1.3 Shape analysis: extraction of shape hierarchies*

Since any set of non-overlapping polygons can be used as a surface, we can also derive surfaces from other sources such as image analysis. Any image can be converted to greyscale and thresholded at different greyscale levels to derive contour lines, resulting in a set of non-overlapping but possibly nested regions (this can include regions within other regions and also holes within regions). The nesting structure of this resulting set of regions can be analysed, resulting in a tree structure that is inherent to the image. If desired, the resulting shapes can be further partitioned into more localised subregions, whilst maintaining that all subregions of an original region point to the same node on the tree.

Whilst procedural generation can in theory produce infinite detail, shape analysis can only produce finite levels of detail and zoomability.

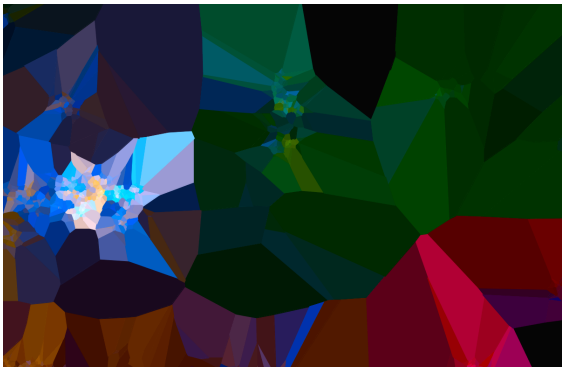

**Figure 3**. Procedurally generated surfaces. Random points were generated using Perlin noise. The points were then assigned to a tree using hierarchical clustering. Colours were assigned to points according to their position in the tree. Voronoi tessellation was used to convert points to surface regions.

Although it has not yet been tried in a musical example we adapted software used for thresholding and edge detection to produce shapes that fit our polygonal surface description format. A trivial extra step is to extract the shape hierarchy from these images and colour surfaces accordingly if required. This allows us to take any image as source material for a surface, and derive the surface polygons and hierarchy from the image, from which the OSC message contents can also be derived. Obviously, as with procedurally generated shapes, the decision about what musical structures will respond to shapes has still to be made.

#### *3.1.4 Derivation of shapes from musical data*

An obvious alternative is to work the other way and derive shapes from the sonic material that will be controlled. A well-known example of such a process is Schwarz's *CataRT* software [6], which draws scatter-plots of granular frames from a corpus of audio recordings using low-level audio features. The user can then play back sound in a feature-driven manner by hovering their mouse over the scatter plot. Hierarchical musical analysis, for example using the methods in Lerdahl and Jackendoff's Generative Theory of Tonal Music [5], can be used to derive trees that describe the relationships between notes in a musical work. Such trees can be mapped to 2D spaces using a variety of methods such as Tree Maps [7]. Similarly, Markov Modelling (for example as used by [8]) can be used to find probabilistic relationships between sequences and an optimisation algorithm can be used to find spatial arrangements between elements whose proximities correspond to the transition probabilities between musical elements. As a final example, modular synth structures represented as trees, for example as patches in MaxMSP, could be abstracted and likewise represented spatially as hierarchies.

We have not attempted any of the above methods at this stage in the research but these possibilities provide a range of alternative approaches to constructing surfaces that are made possible by our first prototypes.

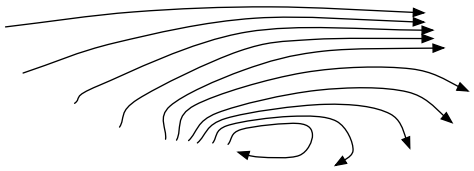

**Figure 4**. Example of interpolating between different paths through a 2D surface.

### 3.2 Methods for traversing surfaces and tree structures

Another motivation for creating surfaces representing musical structure is to define appropriate environments in which automated and generative processes could act to produce semi-composed musical output. For example, a random walk around a Euler lattice is likely to sound more 'musical' [2], at least by tonal standards, than one along a chromatic scale. Likewise, a random traversal of a tree structure results in a kind of output distribution in which closely related states are more likely to occur close together than more distantly related states, but in which sudden leaps can occur. This kind of output can be used to get the effect of self-organised criticality or correspond to observed patterns in musical structure. As such the structuring of the space is a way of incorporating compositional decisions into an instrument, allowing simpler performance processes to act upon structures with meaningful results.

#### *3.2.1 Interpolating and drifting playback agents*

The most obvious playback mechanism is to record a user's actions and repeat them, in a loop for example. In its literal form this has no greater potential than any other form of recording. However, the 2D surface allows forms of path interpolation that would not be possible in 1D. Figure 4 shows an example of a path interpolation that could be used to create a recurring sequence with drifting variation. In such a context the user could specify keyframe paths and set an interpolation going.

#### *3.2.2 Swarms and locally adaptive agents*

Another more generative approach involves 'agents' that traverse the surface based on some kind of interactive behaviour. A continuous 2D space provides a more intuitive environment in which to think about agent behaviour than an abstract musical structure, and agents inhabiting the 2D space of the surface on a tablet can be easily observed and interacted with. Two approaches to designing agents for such spaces are swarms and locally adaptive agents, loosely modelled on Braitenberg vehicles.

Braitenberg vehicles are simple mobile robots (or software simulations of robots) in which two light sensors are connected to two wheel-motors [9]. Depending on whether the connections are parallel or cross-connections, the vehicles behave as either light followers or light avoiders. In a similar way, we can design software agents that travel around surface spaces using virtual sensors to determine their direction of travel. Virtual sensors can use informa-

tion drawn from the surface structure, or in a more complex mode, could be listening to the resulting music.

A swarm of agents provides a simple way to get interesting varying polyphonic structures. Swarms have been used in a range of musical applications, most notably by Blackwell and Young, who used swarms of agents in a parametric musical space to create the effect of self-organised group improvisation [10].

## 4. APPLICATIONS

Our initial research has looked at the most basic examples of interpreting tree structures as sound, focusing on both direct sound generation and the more abstract manipulation of musical elements being played back by the host computer. This work is at an early stage and we accept that it is not presently clear what kinds of mappings from surfaces to musical structures will be of greatest interest. The system in its current state is presented as a framework for exploring creative musical uses based on the design outlined above. Whilst tools such as TouchOSC cater very simple relations between controller and controlled elements, our system is focused on a more involved integration between structure and music. A guiding concept is that the 2D space of the surface, divided into regions, maps to a musical state space, with a region corresponding to a musical state.

From our procedurally generated surfaces, we derived trees, as described above, that can also be read into the recipient music software (the recipient software does not need to know about the surface itself, only the hierarchical relationships between nodes). OSC messages from the tablet to the host computer then identify tree nodes on a per-voice basis such that the recipient software can track where each voice is in the tree and apply transitions as necessary.

A granular sampler was used for playback, and the position of nodes in the tree hierarchy was mapped to control of the sound being played back. Each node was sent over OSC as a path from the tree's root to the node. For example, region 12 in Figure 1 would be sent as the array {1,2,4,9,12}. Since the recipient software is assumed to contain its own representation of the tree this is not strictly necessary, but assuming message density to be minimal this approach is used to provide useful redundancy, e.g., to confirm that the tree is correct or to work even when the tree is not known by the recipient (in the future these long redundant messages may be optional). From the node path, top-level distinctions, i.e., branches just below the root of the tree, such as the number 2 in the sequence {1,2,4,9,12}, are used to choose between different sounds from a bank of pre-buffered sound files. Then following the path from the root down the tree, subsequent distinctions were used to modify the sound in less and less significant ways, beginning with register, then pitch class (based on the circle of fifths), followed by coarse and then finer control of grain size, grain interval and grain rate. The purpose of this mapping is that closer regions in the surface tree representation (those with more similar colours in Figure 3) will play sounds that sound more similar. As you cross a major colour boundary you will hear a more significant change in sound. Although the parameters and the surfaces are randomly generated in this initial experiment, these essential similarity relationships are maintained.

Experiments performing with the system indicated that exploration of the space through zooming, panning and playing was essentially intuitive, with the potential to exploit visual memory to allow fast access to electronic music structures. Whilst the visual memory of structures was basically viable, the method for procedurally generating surfaces was clearly too bland, so surface regions weren't especially memorable. We intend to address this by exploring the use of existing images as source data, as discussed in Section 3.1.3, or using alternative procedural processes. In further work we hope to show that visually memorable surface layouts with consistent mappings to sound can then result in easily learnable performance environments.

### 4.1 Automatic derivation of surfaces from DAW data

Work is underway to adaptively derive surfaces from a variety of existing musical systems. A popular creative electronic music tool is Ableton Live. Live allows realtime multitrack performance through the selection of *clips*, small blocks of music such as audio loops or MIDI sequences. Clips can be triggered using a range of MIDI devices such as the Novation Launchpad. We propose to apply automated analysis to existing Live projects (a "project" is a single document consisting of a number of tracks within which clips can be selected) from which control surfaces could be derived.

### 4.2 Shapes as program code: spatial live coding

An exciting possibility is that the paradigm of shaped-based control of hierarchical sequences can extend to an open-ended programming technique, given the right procedural or design techniques for creating structures that relate to program code. For example, the novice programming tool *Scratch* [8] allows users to construct program code using drag and drop operations on shapes representing blocks of code. The experimental hybrid coding/patching environment *Field* [9] uses the concept of an interactive GUI block which can be coded to respond to user interaction such as mouse-overs. Following our surface paradigm, polygons would represent software objects which would respond to *enter* and *exit* commands, by, for example creating and destroying objects in a signal path. Nested polygons would be handed an environment on entry, consisting of objects created by their enclosing polygons. Procedural methods would enable the generation of vast zoomable spaces of generative possibilities that could be explored interactively.

## 5. DISCUSSION

In this paper we have presented a general approach to creating musical controllers based on the arrangement of polygons in a 2D surface. The purpose of this paper has been

[8] http://scratch.mit.edu/

[9] http://openendedgroup.com/field

to outline the design thinking behind this approach, describe our implementation of the system and discuss our initial reflections using the system musically. Our view is that the software we have started to develop for this purpose will have general applicability to a range of contexts and offers an original and potentially valuable approach to using multi-touch tablets in musical performance, but that our first prototype leaves many open questions about the best design and musical use, which we hope to answer in future experimentation. Our immediate aim is to finalise publicly available apps for loading and using surfaces on Android and iOS devices, and sending OSC messages to a host computer running music software. A general purpose file format for representing surfaces can then be used to load surfaces onto apps but also the corresponding trees into the recipient programs. Whist the app can work as a standalone player/controller, specific standalone tools would be able to generate surfaces and corresponding musical playback systems in the ways proposed in this paper (and in other ways). In addition, different playback agents could also be created as plug-ins for the tablet controllers.

**Acknowledgments**

This project was supported by a residency from STEIM, Amsterdam (http://www.steim.nl), and associated performance at the Sonic Acts Festival, Amsterdam, February 2012.

## 6. REFERENCES

[1] J. McCormack, P. McIlwain, A. Lane, and A. Dorin, "Generative composition with nodal," in *Workshop on Music and Artificial Life*, E. Miranda, Ed., Lisbon, Portugal, 2007.

[2] S. Maupin, D. Gerhard, and B. Park, "Isomorphic tessellations for musical keyboards," in *Proceedings of the 8th Sound and Music Computing Conference*, S. Zanolla, F. Avanzini, S. Canazza, and A. de Götzen, Eds., 2011, pp. 471–478.

[3] D. Norman, *The Psychology of Everyday Things*. Basic Books, 1988.

[4] D. Saffer, *Designing for Interaction: Creating Innovative Applications and Devices*. New Riders, 2009.

[5] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*. MIT Press, 1983.

[6] D. Schwarz, G. Beller, B. Verbrug, and S. Britton, "Real-time corpus-based concatenative synthesis with catart," in *Proceedings of 9th International Conference on Digital Audio Effects*, 2006.

[7] B. Johnson and B. Shneiderman, "Tree-maps: a space-filling approach to the visualization of hierarchical information structures," in *Visualization, 1991. Visualization '91, Proceedings., IEEE Conference on*, oct 1991, pp. 284 –291.

[8] F. Pachet, "Beyond the cybernetic jam fantasy: The continuator," *IEEE Computer Graphics and Applications*, vol. 24, no. 1, pp. 31–35, 2004.

[9] V. Braitenberg, *Vehicles: Experiments in synthetic psychology*. MIT Press, 1986.

[10] T. Blackwell and M. Young, "Self-organised music," *Organised Sound*, vol. 9, no. 2, pp. 137–150, 2004.