Real-time Algorithmic Composition with a Tabletop Musical Interface - A First Prototype and Performance

Cárthach Ó Nuanáin
Music Technology Group
Universitat Pompeu Fabra - Barcelona, Spain
carthach.onuanain@upf.edu

Liam O' Sullivan
Dept. of Electronic and Electrical Engineering
Trinity College Dublin - Dublin, Ireland
lmosulli@tcd.ie

ABSTRACT
Algorithmic composition is the creation of music using algorithms, or more specifically, computer programming. Real-time Algorithmic Composition seeks to extend this compositional technique to real-time scenarios like concert performances or interactive installations. As such, a more novel mode of interaction between the composer and the system is needed for effective expression and parameter control. We summarise the design and implementation of a Real-time Algorithmic Composition system that employs a tabletop musical interface for input control. Several algorithms developed for real-time performance are discussed. The application and inclusion of the tabletop interface is then outlined and evaluated.

Categories and Subject Descriptors
H.5.5 [**Information Systems**]: Sound and Music Computing—*Methodologies and techniques*

Keywords
Computer Music, Algorithmic Composition, Interfaces, Tabletop, Tangible

1. INTRODUCTION
Algorithmic Composition, the creation of music using computer algorithms, has for a longtime been an *offline* activity. The composer typically expresses instructions with computer code, issues the instructions to a computer and receives the final output as a rendered score or audio (using synthesis and/or sampling). Today with the proliferation of high-speed computers and laptops, composers are afforded the opportunity to interact with their algorithms and compositions in real-time and by extension, in a live performance capacity. This brings with it its own new, unique set of challenges. Which algorithmic techniques are suitable for a real-time context? How does the composer interact with algorithmic parameters?

So-called New Interfaces for Musical Expression explore the application of novel new controllers for interacting with computer music systems. It is common to see production environments augmented by a wide variety of commercially available music controllers that extend and complement the level of control offered by the standard music keyboard controller. Complex modes of interaction benefit musicians and audiences alike; the musician can communicate their intent more expressively, while the audience experiences a more dynamic and engaging performance than one where the musician is static behind a laptop.

This paper documents research that explores real-time algorithmic composition with an emphasis on its interaction with interfaces, culminating in the development of a software system titled *ReacTacT*. Three algorithms that demonstrate algorithmic composition applied to a real-time context were implemented. We examine these in detail then describe how a tabletop musical interface was incorporated through consideration for effective parameter mapping. The challenges faced in evaluating such a system in comparison to traditional synthesiser applications are highlighted.

2. STATE OF THE ART
2.1 Real-time Algorithmic Composition
Brian Eno popularised the term “Generative Music” when he released his album *Generative Music 1* in 1996. Collaborating with software company SSEYO, he produced the Koan system (now called Noatikl) to create ”music that is ever-different and changing” [5]. Interestingly, the work was released not as a recording but as a set of instructions for the Koan player, a component of the Koan system itself. Eno preceded many current artists who have adapted similar multimedia approaches to releasing music [11], [7].

In the classical world, Austrian composer Karlheinz Essl is regarded as one of the chief proponents of real-time algorithmic composition. He has developed an extensive library of functions for the Max environment known as the Real Time Composition Library (RTC-Lib) [6]. Essl gives frequent performances with his software, using simple MIDI controllers to interact with and direct the course of the performance.

A more concrete manifestation and extension of algorithmic composition today is the fast-growing culture of ”live coding”, whereby the performer develops and iterates on computer programs that create music in a live performance situation. It is typical for the code itself to be given strong focus by means of video projection of the source code being edited during the performance. This can often be attributed to the very nature of live coding, which requires the performer to remain fixed at their computer keyboard. Some visual feedback that gives context and information on what the performer is doing helps the audience understand the performance more. For a more comprehensive overview of live coding the authors recommend [3].

2.2 The Reactable Tangible User Interface

The Reactable Tangible User Interface is an electronic music instrument that operates by the user moving tangible objects or *fiducials* on an illuminated tabletop surface [12]. It is modelled heavily on the modular "patching" paradigm present in analog synthesisers. Function generators such as sine wave oscillators are connected virtually to filters and effects. It emphasises collaboration and experimentation, and is intended as much for the public space as well as the concert stage. Novices and non-music people can explore sound creation in a fun and interactive manner; established artists such as Bjork can tailor the instrument to suit their specific needs.

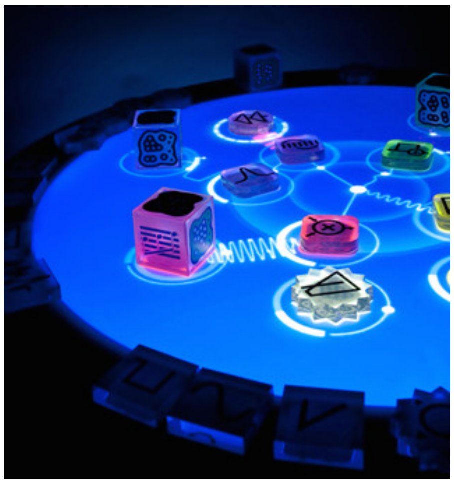

Figure 1: Reactable Tangible User Interface

2.3 Real-time Composition and Tabletop Interfaces

At the time that this research was conducted, the authors were only aware of one other system that employed a Table-based Tangible User Interface for the goal of interacting with real-time software for composition. Xenakis (named after the influential Greek composer and architext Iannis Xenakis), adopts a similar approach to the Reactable in the development of a tabletop interface [2] for music. It uses the relative distances of fiducial markers on the table to define a musical sequencer built on Markov chaining probabilities.

Our approach differed from the Xenakis project in two ways. Firstly, Xenakis is intended for the general public and as such presents a more simplified environment that is more easily accessible for novice users. Our system is intended for the individual composer who will learn and build on it over time. Secondly, Xenakis presents one algorithmic technique using probabilities while we offer a more modular system that includes three algorithms with the intention of expanding with more.

3. IMPLEMENTATION

This section details the implementation of the ReacTacT system. Firstly the algorithmic techniques that were developed for composition are outlined. Secondly the building of the tangible user interface is explained. Finally, aspects of the parameter mapping are explored. The Max/MSP programming environment was used to develop the system, with some algorithms implemented in other languages but interpreted or hosted inside of Max through external objects. Max also served to "glue" the input from the interface to the algorithms and the output from the algorithms to MIDI capable samplers and synthesisers. This overview of this architecture is laid out in Figure 1.

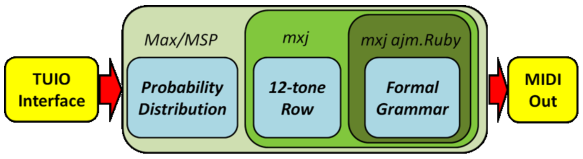

Figure 2: Max/MSP Algorithmic Architecture

3.1 Algorithms

3.1.1 Stochastic Music

A *stochastic process* is a term used in statistics to refer to a random process that obeys some *probability distribution function* [18]. In terms of algorithmic composition it typically implies music created using indeterminacy and chance behaviours.

Stochastic music can be implemented simply and effectively in Max through the use of probability distribution tables. The table object allows graphical edition of an array of numbers corresponding to particular probability distribution. A bang message to table then instructs the table object to output a random number weighted by this probability distribution. One could, for example, map a probability distribution to each note in the chromatic scale and produce diatonic major/minor scales by providing weighted probabilities for the required notes (and none for notes not part of the scale). Going futher, the scale degrees themselves can be weighted further with the preference for the tonic, subdominant, dominant tones for example.

Rhythm values can be encoded in a similar fashion. Values ranging from semibreves to 16th notes for example, can generated according to a desired distribution of rhythmic activity, with rests included as a "ghost" pitch. For our implementation, two table objects are used to output weighted pitches and rhythm values. Figure 3 overleaf shows an example of a scale of pitches mapped to MIDI output in Max/MSP.

3.1.2 Formal Grammars

This section introduces formal grammars as an algorithmic technique inspired by studies in language processing. We outline a method of combining evolutionary algorithms and formal grammars to generate phrases of tonal music in real-time, with composer interaction through the tabletop interface.

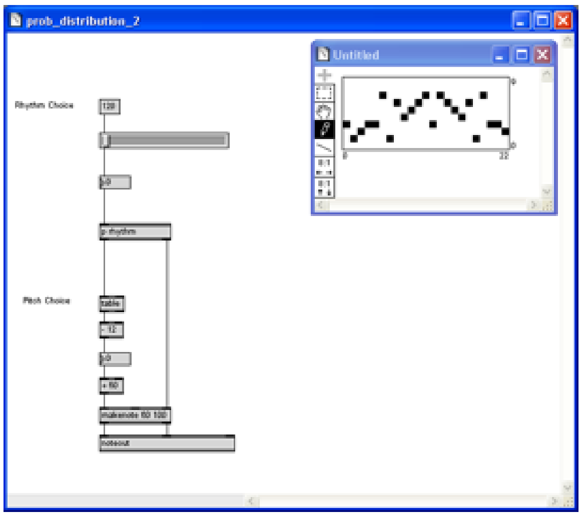

Figure 3: Probability Distribution Tables

Formal grammars, as set out initially by Noam Chomsky, attempt to describe languages mathematically. At its core there exists grammars that define rewrite rules for generating phrases which are syntactically correct and legal in the target language. While formal grammars hold great value in the fields of computer science and linguistics, there has been considerable efforts to apply such techniques to aspects of music. A 1983 book by Lerdahl and Jackendoff provides a comprehensive description of Western music using formal grammars[8].

David Cope has used formalised grammars extensively for automatically creating new pieces of music in a particular style [4]. This has been effectively applied to Bach chorales and Mozart symphonies for example. Cope's approach is very much an offline activity; complete works of music are created and outputted as whole.

Grammatical Evolution is a fairly new programming technique that combines formal grammar theory with genetic algorithms. Genetic algorithms are AI programming techniques that use the biological metaphor of evolution to iterate and find the optimal results for a search problem. Classical features of evolution such as inheritance and mutation are mimicked in software and appraised using a suitable fitness function. The fitness function can be automatic, or in some applications, manually carried out by a human. For subjective search problems such as creating musical output this is the approach typically used - the composer evaluates the output of the genetic algorithm and deems it suitable or unsuitable.

Adam Murrary outlines a way of using grammatical evolution to generate musical phrases according to a formalised grammar [15]. It uses a library for the Ruby programming language called DRP (Directed Ruby Programming) [REF]. Using a Ruby interpreter embedded in a Max external built by the same author, the algorithm can be used to generate streams of MIDI notes. We have adopted this approach for this project, below shows a portion of the grammar used:-

```
#Top Level
max_depth 2
def start; "#{scale}" end
def start; "#{scale} #{start}" end

#Scale Weighting
weight 0
def scale; "#{major}" end
weight 0
def scale; "#{minor}" end
weight 2
def scale; "#{majpent}" end
weight 1
def scale; "#{minpent}" end

...

#majpent
max_depth 4..8
def majpent; "#{majpentn} #{majpent}" end
def majpentn; "#{d}#{majpenti}" end
weight 3
def majpenti; "0" end
def majpenti; "7" end
weight 2
def majpenti; "2" end
def majpenti; "9" end
weight 1
def majpenti; "4" end

...
```

At the top level we define the start of our grammar as consisting of one or more generations of a "scale" phrase. The possible scales in our example include major, minor and the corresponding pentatonics. If you consider the grammar in terms of a syntax tree, the leaves contain the step value of the notes themselves as offsets from a root. Hence the major pentatonic includes the five offsets 0, 2, 4, 7, 9 with the semitone steps from the major scale omitted.

Weightings are applied to nodes to affect the likelihood of that rewrite rule being triggered. The max depth instruction defines the maximum amount of permissible rewrite rule triggering. In our grammar we define very shallow depths of rewrite rules that heavily weight major pentatonics over minor pentatonics. The result of this is the generation of very short pentatonic phrases that are repeated until new instances are triggered. This creates heavily repetitive tonal phrases not unlike the minimalist stylings of composers Steve Reich and Philip Glass.

3.1.3 *Hybrid Algorithms and Serialism*

This section describes an approach of combining stochastic processes with traditionally deterministic composition methods like twelve-tone technique, which was developed by Arnold Schoenberg in the early part of the 20th century. It can be considered a reaction to the practice of atonal music that preceded it, which Schoenberg viewed as too chaotic [19]. The composer proceeds by deciding on an initial twelve-tone row - an explicit ordering of each note in the chromatic scale [18]. Variation and contrast is achieved through a set of discreet modifications to the initial row (known as the *prime* that includes:

- Retrograde - Prime row ordering is reversed

- Inversion - Prime row is inverted

- Retrograde - Prime row is reversed and inverted

Finally, the prime row and all its modifiers can be transposed to every other starting note. It is common to represent all these possibilities in a table known as a twelve-tone matrix.

Serialism, as spearheaded by Schnoenberg's student Anton Webern, can be considered a totalist continuation of twelve-tone technique where other musical dimensions such as timbre and dynamics are "serialised" in a similar fashion. It should be evident from this brief discussion that the tightly parametrised and formalised quality of twelve-tone technique and serialism lends itself well to algorithmic composition. We implemented a simple twelve-tone row generator as a Java-based external for Max/MSP. The *toneRow* object can receive an initial tone row from the composer or generate a new one randomly. When it receives a bang message it outputs the tone row with the possibility of a transformation applied. The tone row pitches are then fed to MIDI objects for output to sound generators.

Twelve-tone technique and serialism can be considered a deterministic style of composition, in that the output is predicable. Indeed, as alluded to earlier, Schoenberg sought to bring order to a previously considered chaotic practice. By allowing the possibility of creating an initial tone row randomly however, we have introduced some element of chance to the results.

3.2 Reactable

The complete Reactable system is a commercial product developed and sold by Reactable Systems. The commercial product includes the hardware setup, a complete sound engine capable of synthesis, sampling and step-sequencing, and a visual engine for projection feedback onto the table. As seen in Figure 1, the visual engine is for many the most compelling, striking feature on first encounter. The synthesis and visual engine is only available when the complete Reactable system is purchased but fortunately the computer vision framework *reacTIVision* that tracks the markers is a available for free as open-source software[13]. A useful aid when developing software using reactTIVision is the simulator, which allows the developer to simulate a physical Reactable session with the mouse and keyboard on screen.

Figure 4 shows the hardware setup of a typical Reactable system. Fiducial markers are manipulated on top of a glass surface. Below, a digital camera tracks the movement of the markers while a projector projects visuals onto the surface. The digital camera operates in the infra-red spectrum in conjunction with an infra-red light source so that the projector can operate in the visual spectrum independently.

[1]http://reactivision.sourceforge.net/

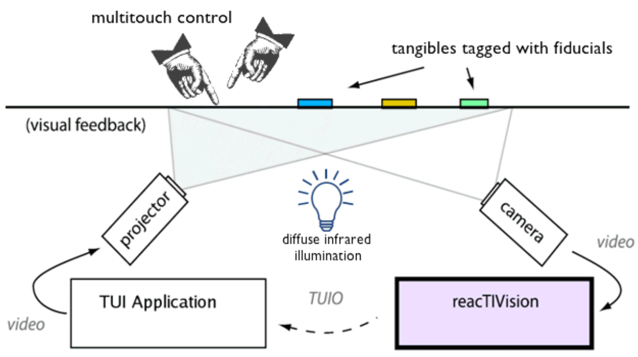

Figure 4: Reactable Hardware Setup[1]

The focus of this project was the integration of a tabletop interface with real-time algorithmic composition, as such, the added complexity of back projection was not considered due to the workload and timescale. Figure 5 shows the final table in a performance setting without the back projection.

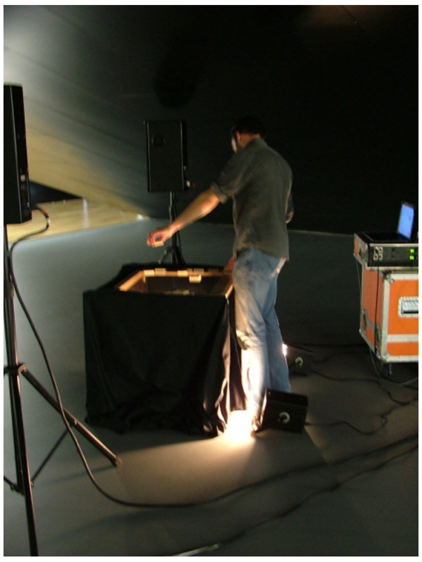

Figure 5: Tabletop Hardware Setup

Even though back projection to the table was not included for the project, a program for generating graphical visuals was implemented in the Processing language with the intention of projecting live onto a screen during performance, primarily for the audience's experience. Simple shapes, animations and connections correspond to various algorithmic activity on the board, in patterns somewhat inspired by artist William Kandinsky's "Visual Music" style abstract art (Figure 6).

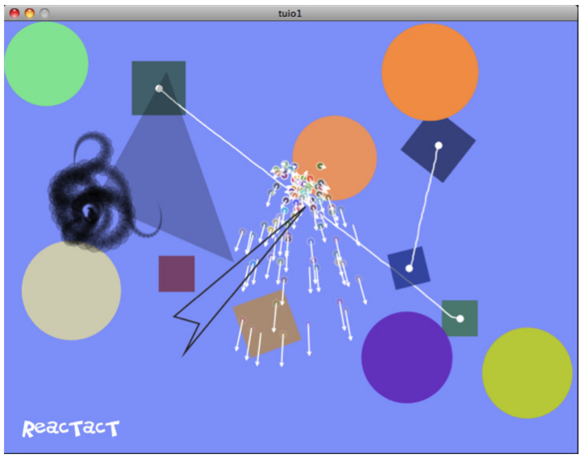

Figure 6: Processing Visuals

3.3 Parameter Mapping

As each algorithm was being implemented, the possible modes of interaction between the algorithms and the tabletop interface were considered carefully. The incorporation of any controller interface to the computer becomes a task of parameter mapping: the connection between human gesture and software response. Much research and literature has been conducted on the importance of parameter mapping and user interaction in musical interfaces.

Hunt et al. in particular explores mapping in developing effective electronic instruments [10]. They distinguish between one-to-one, one-to-many, and many-to-many as potential mapping schemes between interface and software. It is argued that a good scheme is one that employs a complex strategy, using many-to-many connections and possibly multiple layers of mapping.

The challenge of parameter mapping for real-time composition is a very different one to that of say, a modular synthesiser. When a musician presses down the middle C key or adjusts the filter of a Moog for example, a well defined event results in an immediately recognisable response. The difference can be summarised in two different level of interaction. Wanderley identifies such low-level manipulations of parameters like pitch and timbre as *note level* or micro level [20]. With real-time composition however, we prefer to describe the level of interaction as macro or *score-evel.* The composer is interacting with software systems that make compositions rather than generate sounds.

Our chosen parameter schemes came from experimentation with the algorithms as we developed them. The overarching motive is to provide as much control over compositional algorithms as possible but in a high-level and abstract way. Composers should direct the overall direction and mood of a piece, rather than worrying about lower-level things like instantaneous pitches and note lengths.

3.3.1 Algorithm Mappings

For our first and conceptually most simple algorithm - the probability distributions - the solution seemed similarly simple and elegant. Two fiducial markers are assigned to the melody and rhythm tables respectively. Just as the probability distribution is drawn in software with the mouse, it is expressed with a fiducial marker directly on the table. Music is generated and iterated on continuously, the only drawback being the lack of visual feedback - unless the composer is looking at the screen, they don't know the status of their distribution visually and have to rely on auditory feedback alone.

In the case of formal grammars algorithm, it was intended initially that the composer would play the role of providing a kind of real-time fitness function for the generation of preferred musical phrases. This proved difficult as many of the features of the DRP library did not lend itself well to real-time interaction. Maximum depths and weighting factors were "fixed" into the script at interpretation time by the Max external. In the final implementation the scheme outlined in Table 1 was chosen for mapping between the interface and algorithm.

Table 1: Formal Grammar Mapping

Interface Parameter	Algorithm Parameter
X Axis Position	Reference Note (Default - C4)
Y Axis Position	Rhythmic Unit (crotchet, quaver etc.)
Angle of Rotation	MIDI Velocity (Loudness)
Marker at table centre	Triggers a new note from the grammar system

When dealing with the twelve-tone algorithm, the final mapping scheme that was decided upon was very much influenced by the notion of a conductor directing a string quartet, since this was the sound set chosen for performance (Figure 7). The whole quartet can be controlled as a single entity by placing the marker in the centre of the table.The four rotations of the square fiducial selects tone-row transformations to be applied. Control of individual instruments in the quartet is achieved by placing the marker in the appropriate portion of the table. As you view the table from above, the top portion targets Violin 1 with the bottom portion targeting Violin 2. The left and right sides address the viola and cello respectively. This gives simple but powerful control over the entire ensemble offering the possibility for interesting experimentations and combinations.

3.4 Evaluation and Discussion

Evaluation for the project was conducted with two motivations in mind. Firstly the objective value of the tabletop interface as a controller for interacting with computer music software was explored, drawing from some strategies prevalent in the field of HCI (Human Computer Interaction). Secondly, the more difficult task of subjectively analysing the musical output of the system as it exists holistically was investigated.

3.4.1 Interface Evaluation

It is important to reiterate our previous observation that new interfaces and controllers tend to target note-level sound generating instruments like synthesisers. Consequently, existing methods of evaluation for such interfaces examine the usability for that purpose and not higher-level applications like real-time algorithmic composition. Still, it is crucial to

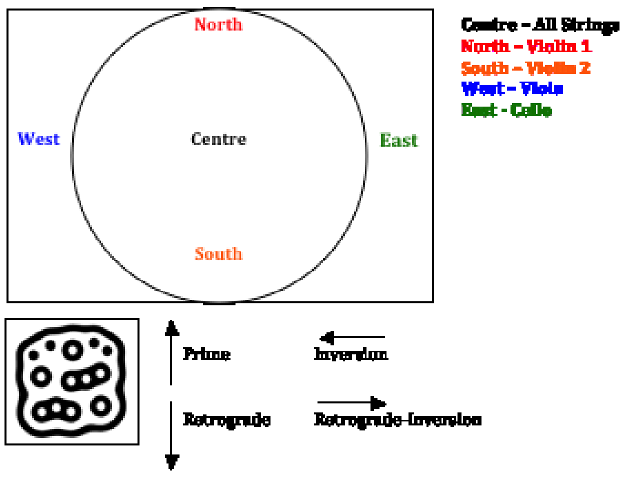

Figure 7: Serial Algorithm Parameter Mapping

survey the approaches used to determine whether any are adaptable in our case.

In evaluating interfaces for musical expression, strategies are frequently adopted from the field of HCI (Human Computer Interaction) such as proposed by [20]. Many HCI tests centre around task-driven experiments, such as the time and ease of the system to reach a well-defined target. In traditional HCI this may include, for example, manipulating a Graphical User Interface (GUI) with a mouse controller in order to move a file icon somewhere.

Kiefer et al. [14] takes Wanderley's proposal and applies it to the evaluation of the Wiimote as a musical controller. Four musical tasks are defined for users to carry out. One task for example requires the user to trigger a drum sample in time with a metronome. For comparison, the commercially-developed Roland Handsonic 15 drum controller was also used to carry out the same tasks. Overall, the users had no particular preference for either interface, except for the triggering task where 70% favoured the drum pad. This correlates well with the fact that this interface is designed precisely with percussion in mind. Hsu examines HCI methods of evaluation in the context of a computer improvisation system [9]. They suggest that an evaluation framework should address the usability of an environment for a human improviser and whether the results of the performance are musically interesting for an audience. We can see a clear overlap here with the goals and evaluation methodologies that would apply to our system.

3.4.2 Tabletop Interface Evaluation

In order to determine the relevance of HCI approaches in our system and in real-time composition in general, we conducted a trivial test to determine the ability of the interface in achieving a well-defined compositional task that may be required when composing in real-time [18]. The task was conducted by one of the authors who consequently would be composing and performing with the instrument. Using the probability distribution algorithm, the task required the composer to:

> Create a probability distribution function that corresponds to the scale of C Major (Assuming C (MIDI Note: 60) is the reference note) then choose a crotchet note duration

The tabletop interface was compared against an M-Audio MIDI controller with pitch wheel and the regular computer keyboard and mouse. The time to complete the task with each controller was then recorded with a stopwatch. The table below shows the results:

Table 2: Tabletop Interface Evaluation

Interface	Time
Computer mouse and keyboard	00:14:97
MIDI keyboard/controller	00:07:13
Tabletop Interface	01:12:16

It is evident from the results that the tabletop interface has been outperformed considerably by the other controllers in this task. One might conclude that the interface is not up to the task at all but that would be to miss the bigger picture. The goal is to use the interface as an *exploratory* tool for algorithmic composition, opening up new possibilities of music creation and breaking past habits or patterns. The next section briefly discusses the subjective experience of using a tabletop interface in this light.

3.4.3 Compositional Evaluation

The musical output of the system was evaluated by the authors by performing with in a live capacity, recording music created with it then documenting the process and the subjective impressions in the thesis report. A complete performance using the ReacTacT system was premiered by one of the authors at the Samuel Beckett Theatre in Trinity College in Dublin[17]. A subsequent performance was delivered at the Darklight X Film Festival.[16]. Algorithmic music composed using the system was also featured on the Dublin Electronic Arts Festival compilation CD [1]. Reception and feedback from audiences in each event was favourable and people were interested in learning more about the nature of the system and its design.

4. FUTURE WORK

We were pleased with the overall outcome of this project and wish to pursue some promising avenues for future work that are outlined here.

Visual feedback is an integral element of the tabletop interface and it was unfortunate that we couldn't incorporate it into our system. Luckily, since completing this research, one of the authors has joined the Music Technology Group at Universitat Pompeu Fabra in Barcelona, where the Reactable was originally conceived. It is the intention to update the ReacTacT software to use the fully working Reactable setup with particular focus on improving the visual element using the back projection of the table.

Our system can be considered very much as a *symbolic-based* approach in that we work with symbolic representations of music including MIDI and list structures corresponding to pitch and rhythm. Another method would be work with *content-based* audio information (such as existing sounds and samples) and extract musically meaningful features to manipulate and generate new music from this existing audio. Recent research has been explored that uses flocking algorithms to automatically navigate the parameter space of concatenative/granular synthesisers. This can be extended to the ReacTacT system, where fiducial markers can be used to ”guide”a flock of swarming agents that control parameters of the compositional algorithms.

Finally the challenging, complex task of evaluation needs further consideration. Comparing task completion times was revealed not to be very useful measure of appraisal. It would be worthwhile coming up with new domain specific tasks that address the exploratory and experimental nature of the instrument. Also, as alluded to the in the evaluation section tests were only conducted by the author/composer. More effective evaluations would benefit from extensive user testing with other composers. A more general task-oriented approach where composers give feedback on their experience with different controllers controlling the different algorithms may present more appropriate results.

5. CONCLUSIONS

This paper presented a system for real-time composition that uses a tabletop musical interface inspired by the Reactable for controlling parameters of the algorithms in a novel manner. Several techniques for algorithmic were outlined and the issue of parameter mapping between the interface and each algorithm was explored. Evaluation was revealed to be a difficult, subjective task in such a particular and esoteric system. Finally we suggested some opportunities for future work. Further details about the ReacTacT platform and full source code is available on the author’s website:-
http://www.carthach.tk/projects.html.

6. ACKNOWLEDGMENTS

This research has been partially supported by the EU-funded GiantSteps project (FP7-ICT-2013-10 Grant agreement nr 610591). [2]

7. REFERENCES

[1] DEAF CD 2009. http://deafireland.com/2009/?page_id=16, 2009.

[2] Bischof, M., Conradi, B., and Lachenmaier, P. Xenakis: combining tangible interaction with probability-based musical composition. *Proceedings of the Second International Conference on Tangible and Embedded Interaction*, pages 121–124, 2008.

[3] Collins, N. Generative music and laptop performance. *Contemporary Music Review*, 22(4):67–79, Dec. 2003.

[4] Cope, D. *Computers and Musical Style.* A-R Editions, 1991.

[5] Eno, B. Generative Music. http://intermorphic.com/sseyo/koan/generativemusic1/, 1996.

[6] Essl, K. Real Time Composition Library (RTC-lib).

[7] Guardian, T. Bjork: Biophilia App. http://theguardian.com/music/musicblog/2011/jul/20/bjork-biophilia-app, 2011.

[8] Hantz, E. : A Generative Theory of Tonal Music . Fred Lerdahl , Ray Jackendoff, 1985.

[9] Hsu, W. and Sosnick, M. Evaluating interactive music systems: An HCI approach. *Proceedings of International Conference on New Interfaces for Musical Expression*, pages 25–28, 2009.

[10] Hunt, A., Wanderley, M., and Paradis, M. The importance of parameter mapping in electronic instrument design. *Journal of New Music Research*, pages 1–6, 2003.

[11] Intel Research. Internet Killed the Video Star: Mobile Apps Redefine the Album Experience. http://iq.intel.com/internet-killed-the-video-star.

[12] Jordà, S., Kaltenbrunner, M., Geiger, G., and Alonso, M. The reacTable: a tangible tabletop musical instrument and collaborative workbench. *ACM SIGGRAPH 2006 Sketches*, 2006.

[13] Kaltenbrunner, M. and Bencina, R. reacTIVision: a computer-vision framework for table-based tangible interaction. *Proceedings of the Second International Conference on Tangible and Embedded Interaction*, 2007.

[14] Kiefer, C., Collins, N., and Fitzpatrick, G. Evaluating the wiimote as a musical controller. *Proceedings of the International Computer Music Conference*, 2008.

[15] Murray, A. Musical Grammar Evolution In Max Via Ruby. http://compusition.com/news/2008/4/23.

[16] Ó Nuanáin, C. Darklight X - Festival Performance. https://www.youtube.com/watch?v=c6c5ixQTSEc, 2009.

[17] Ó Nuanáin, C. Fresh Intelligence - A Study in Realtime Composition with the ReacTable Tangible User Interface. http://vimeo.com/7392845, 2009.

[18] Ó Nuanáin, C. The ReacTable as a Platform for Real-time Algorithmic Composition. Master’s thesis, Trinity College Dublin, 2009.

[19] Román, B. D. Twelve-Tone Technique : A Quick Reference. *Musiké, The online journal of the Conservatory of Music of Puerto Rico*, 2008.

[20] Wanderley, M. M. and Orio, N. Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI. *Computer Music Journal*, 26(3):62–76, Sept. 2002.

[2]http://mtg.upf.edu/projects/giantsteps