Proceedings of the

Improvisational Creativity Workshop

SensiLab

Australian Government
Australian Research Council

PRATO WORKSHOP		WEDNESDAY 19th July	THURSDAY 20th July	FRIDAY 21st July	
MORNING	09:00 - 09:30	Welcome and Introductions	Improv exercise #2 – theatre sports	Improv exercise #3 – sound	
	09:30 - 10:00		The Future of Creativity Mark d'Inverno	Interfaces Panel Justyna Ausareny, Peter Beyls, Shelly Knotts, Eleonora Oreggia, Amble Skuse	
	10:00 - 10:30	Improv exercise #1 – image making			
	10:30 - 11:00	BREAK	Hands on with Tech Resources		
	11:00 - 11:30	Introduction to Tech Resources	BREAK	BREAK	
	11:30 - 12:00		Writing - group breakout	Writing – group breakout	
	12:00 - 12:30	Discussion: The big challenges in improvising with computers			
	12:30 - 13:30	LUNCH	LUNCH	LUNCH	
AFTERNOON	13:30 - 14:00	Paper presentations I	Regroup and report on progress	Creative development #3 refinement	I-CubeX & Bela maker session
	14:00 - 14:30		Paper presentations II		
	14:30 - 15:00	Group distillation of questions			
	15:00 - 15:30	BREAK	BREAK	BREAK	
	15:30 - 16:00	Regroup and consolidate topics	Creative development #2 – construction	Performance / Presentation of developed works and reports on progress on papers	
	16:00 - 16:30	Creative development #1 – ideation			
	16:30 - 17:00				
EVENING	17:00 - 17:30	BREAK	Concert in the Garden Stefano Kalonaris, xname, Matthew Yee-King & Mark d'Inverno, Henning Berg, Stanislav Nikolov, Joanne Armitage & Shelly Knotts	Wrap-up	
	17:30 - 18:00			Participant Jam and drinks	
	18:00 - 18:30				
	18:30 - 19:00	TRIP TO FLORENCE			
	19:00 - 19:30			DINNER	
	19:30 - 20:00		DINNER		
	20:00 - 20:30				

SESSION TYPES

Orange — Improvisation Exercises

We start each morning with a warm-up improvisation session, drawing upon a different creative domain each day. The idea of these sessions is to get everyone improvising in a fun and low-stress environment, to get to know the other participants, and to get the creative juices flowing.

Green — Hands-on Creative Development

There's nothing like getting stuck into a project to clarify ideas, and make the abstract concrete. Running across the three days of the workshop we will work in groups towards an improvised performance on the last day.

Blue — Collaborative Writing

One of the goals of the workshop is to identify a handful of key challenges relating to improvisational creativity, interface design and AI; and to start addressing them. We aim to form groups of interested participants to work together on an issue, and create an outline of a research paper on the topic. We have arranged a special issue of the journal *Digital Creativity* as a publication outlet for these collaborative papers (subject to their usual peer review process), which is slated for early 2018.

Red — Provocations and Ruminations

Each morning we have a presenter or panel discuss an idea intended to provoke thought, spark ideas, and canvas different aspects of improvisational creativity. Discussion and dissension are encouraged!

Pink — Paper Presentations

Short presentations by participants of their research and/or practice in improvisational creativity.

Purple — Performances

A formal concert of selected works on Thursday night, a less formal performance of the group projects developed over the workshop on Friday afternoon, and an informal Jam to finish off the workshop on Friday night.

Wednesday 19th July

0900-0915 Registration

0915-1000 Welcome and Introductions

1000-1030 Improv Exercise: Image Making

Collaborative improvised drawing! A fast-paced group exercise, somewhere between pictionary, charades and pass-the-message. Each group has a large piece of paper. One person draws an abstract ‘squiggle’ and the next completes it into a drawing of something.

1030-1100 BREAK

1100-1200 Tech Workshop: Intro to Tech Resources

Technical introduction to available tools and resources for creative development sessions.

1200-1230 Improvising with computers – the big questions and challenges

Introduction to several key challenges for the field followed by whole group discussion and mind-mapping.

1230-1330 LUNCH

1330-1430 Papers I [15mins per paper + 5mins questions/handover]

Improvisation as Speculative Computing - *Peter Beyls*
Technology for Better Improvisation when Teaching Coding - *Matthew Yee-King*
Vibrations in Improvisation - *Joanne Armitage*

1430-1500 Group Discussion: Distillation of questions

Break into smaller groups each discussing the questions/challenges in greater depth.

1500-1530 BREAK

1530-1600 Discussion: Regroup and Consolidate Topics

Small groups present question/challenges discussions to whole group.

1600-1700 Creative Development I: Ideation

Form new groups around the outcomes of question discussions and start developing ideas for a self-contained creative project to work on.

18:30 TRIP to Piazzale Michelangelo, Florence

Make your way to Florence where we will meet at Piazzale Michelangelo for spectacular views of the sunset over Florence. Followed by drinks and food at Vip's Bar (Viale Giuseppe Poggi, 7, 50125 Firenze, Italy)
Suggested trains leave Prato Centrale at 18:29 and 18:42
Directions: https://goo.gl/2ipp4j

Thursday 20th July

0900- 0930 Improv Exercise: Theatre Sports

A practical exploration of theatre improvisation through movement, voice and miscellaneous objects.

0930-1030 Talk: Mark d'Inverno *The Future of Creativity*

There is a constant and sustained buzz around the word "creativity". Reference to it has spread prolifically within and beyond academia, being associated with novelty, value, imagination and innovation. However, I will argue in this talk that the word has been used so much, and in so many different ways, that it has become devalued. Indeed in many cases the term is used to mean little more than that which we approve of.

I believe we need alternative terminology, to regain clarity and currency for the kinds of activity which we want to encourage in schools, universities and society at large. To this end I want to introduce "creative activity" as an alternative to "creativity". By considering recent research and teaching innovations at Goldsmiths, I will look to answer the following key questions based on my experience as a musician, lecturer, and researcher: (i) *what is creative activity?* (ii) *what pedagogy should we use for teaching creative activity?* (iii) *what can we do with current Artificial Intelligence research to build new technologies that can inspire creative activity?* I will do this by specifically focusing on Music improvisation as a form of human creative activity

1030-1100 Tech Workshop: Getting your hands dirty

Hands on session with embedded technology for audiovisual improvisation

1100-1130 BREAK

1130-1230 Writing: Group Breakout

Split into groups and start writing papers.

1230-1330 LUNCH

1330-1400 Discussion: Regroup and report on progress

Writing groups feedback to whole group on progress made in morning session.

1400-1500 Papers II

Tango^2: Software for Computer-Human Improvisation - *Henning Berg*
Comp. Method for Periodic Nonlinear MIDI Seq. and Perc. Improv. - *Clayton Kardas*
Evaluating Improvisational Interfaces - *Toby Gifford*

1500-1530 BREAK

1530-1700 Creative Development II: Construction

Hacking session. Practical development of ideas from ideation session.

1700-1930 Concert

17:30 Dory: a forgetful artificial free improviser - *Stefano Kalonaris*

17:45 Imaginary Flute - *xname*

18:00 Improvised Duet with Audience Visualisation - *Mark d'Inverno, Henning Berg and Jon McCormack*

18:15 Tango^2 - *Henning Berg*

18:30 Neural-Beats: Rhythm and Sound - *Stanislav Nikolov*

18:45 ALGOBABEZ - *Joanne Armitage and Shelly Knotts*

1930-2030 DINNER

Friday 21st July

0900-0930 Improv Exercise: Sound

Algorithmic text-score writing and lo-fi electronic instruments.

0930-1100 Panel and Roundtable: Interfaces

Justyna Ausareny, Peter Beyls, Shelly Knotts, Eleonora Oreggia, Amble Skuse

1100-1130 BREAK

1130-1230 Writing: Group Breakout

Continue working on group writing projects.

1230-1330 LUNCH

1330-1500 Creative Development III: Refinement

Refine creative development projects into working prototypes.

1500-1530 BREAK

1530-1700 Show and Tell. Performing Improvisations and Presenting Progress

Presentation of creative development outcomes and feedback to whole group on progress made in writing sessions.

1730-1930 PARTICIPANT JAM AND DRINKS

2030-2130 DINNER

Performances

Dory: a forgetful artificial free improviser	*Stefano Kalonaris*
Tango[2]	*Henning Berg*
Imaginary Flute	xname (Eleonora Oreggia)
Improvised Duet with Audience Visualisation	*Mark d'Inverno, Henning Berg & Jon McCormack*
ALGOBABEZ	*Joanne Armitage & Shelly Knotts*

Improvisation as Speculative Computing

Peter Beyls
Ear to the Earth
New York, USA
peter@peterbeyls.net

ABSTRACT
We develop a framework towards the characterization of reactive and interactive systems in the context of creative software supporting rewarding human-machine experiences. First principles of coexistence, mutual influence, participation, symbiosis and the maximization of diversity are addressed. The objective is to develop interactive music systems exhibiting at once coherent and unpredictable behavior. Nature itself is considered a source of boundless inspiration.

Author Keywords
improvisation; interaction; interface design

ACM Classification Keywords

Human-centered computing: Human computer interaction; Human-centered computing: Interaction paradigms

Our contribution suggests a comparative study of responsive and interactive systems in the context of creative coding for human-machine improvisation. Open improvisation with machines implies a number of first principles. The principle of coexistence suggests man and machine evolve in a shared biotope possibly sharing mutual objectives in a common effort. According to the principle of mutual influence, by definition, no one is in control; man and machine contribute with equal authority to the subsistence of a reciprocal playground.

A variable floating construction emerges – particular patterns materialize from the expression of internal systemic decision-making and external activation impinging on the system. We are interested in systems offering the capacity to generate at once surprising and coherent behavior. In contrast to merely responsive systems, deep interaction avoids clear one-to-one correlations between human and machine activity. Triggered responses echo human initiative while interactive systems respond autonomously. Perhaps ironically, rewarding interactivity implies autonomy i.e. on the fly creation of appropriate functionality in the face of unpredictable external challenges.

ICW2017, July 19 - 21 2017, Prato, Italy.

Extended instrument models focus on generating complex, augmented morphologies in improvisation with the assumption of a clear correspondence between action and reaction. Machine improvisation based on speculative psychological models aims to reflect aspects of biological life, focusing on spontaneous implicit behavior rather than patterns of explicit design. A wide range of approaches exists between two opposite orientations: purely reactive systems and systems that offer the impression to be alive i.e. apparently equally self-sufficient and warily responsive to external context.

Notions of musical pattern underpin most reactive systems built on algorithms producing complex evolving morphologies in time. In addition, a primarily transparent causal relationship between human input and sonic result offers a ground for straightforward aesthetic appreciation. Contrastingly, interactive systems imply spontaneous behavior as a foundation of complex morphology.

Emergence of complex forms and behavior as characterized by John Holland involves three essential components – translated into the realm of music, as follows: (1) active mechanisms, musical agents containing generators generating perpetual novelty i.e. a virtually infinite number of original and unique musical statements, (2) dynamics and regularities; particular recurrent temporal structures momentarily but clearly perceptible in an evolving musical fabric and (3) hierarchical organization, agents are organized in agencies that themselves become generators at higher levels of organization [1]. Our software adopts these ideas as design guidelines in the understanding that systems should be dynamic, not merely procedural.

Dynamic interactive systems offer strong potential for surprise and anticipation, much like in jazz improvisation; improvisers negotiate in a shared framework driven by uncertainty [2]. In this light, creative software development in general can be characterized as *speculative* computing. Essentially, one implements hypothetical models of processes one

expects to yield complex rewarding (audiovisual) experiences. As the program talks back to the artist, one gets feedback on the aesthetic potential of the instigating idea – so creative programming actively supports aesthetic introspection. In other words, tentative ideas are explored in continuing fashion, through gradual specification of objec-

tives; the focus is mobile and informed by unanticipated response. Note this approach is entirely in line with the notion of free improvisation as a uniquely valued attitude, according to Steve Lacy: "...being on the brink of the unknown and being prepared for the leap (...) if through that leap you find something then it has a value which I don't think can be found in any other way." [3]

With interactive systems featuring emergent behavior, we still expect a cognitive link to appear between user activity and system output, though we avoid trivial connectivity. In practice, a continuous scale of understanding exists correlating human and system activity between two extremes; (1) the system echoes generative interpretations, aspects of a musical context provided by a human performer and (2) the system is totally autonomous and develops its own agenda. The agenda is adaptive to external influence; man and machine are seen as live partners in a common biotope. A dialog-like, conversational approach [4] sits in between; human and machine articulate statements, negotiating and exchanging musical ideas in a common temporal niche.

Previous work explores distributed agencies where agents express social affinities and genetic programming generates appropriate musical functionality on the fly [5]. A recent implementation (called *Pock*) explores a reinforcement-learning algorithm for optimizing musical intimacy in a social setting of a single human performer and one artificial musical agent. The rationale is that we aim to grow a system from scratch, avoiding predefined data structures and complex knowledge representations. The system should learn without prejudice, by gradually building a model of (an essentially unpredictable) human performer seen as a dynamic environment. Pock develops musical policies using an implicit reward function. We infer the inclination of the performer from the observation of two competing motivations: social integration and independent self-expression. For example, policies helpful in reducing the melodic distance between man and machine in case of integration are proportionally rewarded. Lists of policies are dynamically updated and adapt gracefully to external human pressure.

More recent work studies the notion of *social mediation* – how software might optimize the social climate between two performers as to maximize chances for successful performance. We track, analyze and compare changes in physiological signals (such as heartbeats) from two performing bodies and generate arrays of musical policies helpful in minimizing the behavioral gap between both performers. Then, indirect communication between the performers happens through sound, their operational connection is only informed by spontaneous perception of musical patterns. Performers link intimately in a systemic performance loop, influencing systems behavior by reciprocally adapting in a common embodied environment to real-time audio. Embodiment is key to enactment, when musical patterns induce sensory patterns in the brain in leading to physical action [6]. For example, from the viewpoint of the human performer, speculative, intuitive improvisation also implies enacted interpretation based on the ability to turn complex patterns into units that can be dealt with in terms of expressive actions. In conclusion, interactive improvisation blends ideas of introspective speculation, adaptation and, in particular, reward processing.

REFERENCES

1. John Holland. 1998. *Emergence, from Chaos to Order*, Oxford University Press, Oxford, UK.
2. David Borgo. 2005. *Sync or Swarm, Improvising Music in a Complex Age*, Continuum, New York, NY.
3. Derek Bailey. 1980. *Improvisation: Its Nature and Practice*, Moorland Publishing, UK
4. Joel Chadabe. 1989. Interactive composing; an overview, in *The Music Machine*, C Roads (ed.), MIT Press, Cambridge, MA
5. Peter Beyls. 2011. Structural Coupling in a Society of Musical Agents, in: *Artificial Life and Music*, E. Miranda (ed.), A-R Editions, Evanston, WI
6. Marc Leman. 2016. *The Expressive Moment*, MIT Press, Cambridge, MA

Tango^2: Software for Computer-Human Improvisation

Henning Berg
Hochschule für Musik und Tanz Köln
Försterstr. 25, 50825 Köln, Germany
mail@henning-berg.de

ABSTRACT

This paper describes Tango^2, software for Computer-Human Improvisation developed for more than 25 years by Henning Berg. Tango^2 (T^2) listens to an improvising musician, analyses what it hears and plays musical responses which relate directly to the musical input. If the improviser in turn reacts to these answers, a musical loop between the human and the machine can emerge. The way input and reaction correlate and the predictability of Tango's responses can be defined by the user via a setup of improvising environments, called Rooms.

Real-time sampling with knowledge of the musical content behind the samples and MIDI-handling are unified via Tango's own monophonic audio-to-MIDI, time stretching and pitch shifting algorithms. Both audio and MIDI can be used by Tango's modules (e.g. Listeners, Players, Modifiers, Metronomes or Harmony) for input and output. A flexible real time control system allows for internal and external remote control and scaling of most parameters. The free software for Windows7 with all necessary folders, English and German manuals, many example-Rooms and a few videos can be downloaded at www.henning-berg.de.

ACM Classification Keywords

Human-centred computing: Interaction techniques; Applied computing: Performing Arts

Author Keywords

improvisation, computer-human interaction, feedback

TANGO FOR THE ATARI AND TANGO^2

After hearing George Lewis perform with his Voyager at the ICMC 1988 in Cologne, I decided to learn how to program in C and to develop an interactive system for myself. Improvising with a machine was never meant to be a replacement or a practice tool for playing jazz with real people. Instead, it requires a rather different approach and has a quality of its own. During the nineties I played many gigs with my first improvising software Tango for the Atari, published by Steinberg 1991. The band *Tango & Company* with the late English jazz-pianist John Taylor, Tango and myself performed at important festivals in England (Bath) and Germany (e.g. Berlin), played gigs for West German Radio (WDR) and recorded a 'Trio' CD (*Tango & Company 1997*, Jazzhaus Musik). Tango for the Atari was the first program I had written, so due to its limited architecture and many additions, it started to lose some of its former stability and finally also its hardware platform, the Atari. Thus, in 1998 it was time to start over. With Tango^2 I try to incorporate my concert- and programming experience in a software that addresses many of the old, basic ideas in a more flexible and comprehensive way.

BASIC IDEAS

The key feature of Tango^2 is its clear modular design: There are many different software devices (modules), which the user can use and connect in a Room. There are Listener-, Player-, Modifier- and other modules. They can be applied very flexibly to accommodate for simple or complex stimulus-response configurations according to one?s needs.

Particularly important is the integration of MIDI and audio in all parts of the program: T^2 has a built-in audio-to-MIDI module for monophonic instruments such as wind-instruments, voice and strings. This makes it possible to communicate with the program via a MIDI keyboard or directly via a microphone, without an external pitch-to-MIDI device. Additionally, T^2 can not only play MIDI sound generators but also use and modify the audio-signal it received moments ago from the human partner for its responses. The same modules are used for audio and MIDI functions so it makes no difference if T^2 works with MIDI or with audio. This is possible, because T^2 analyses the musical structures behind the samples as if they were MIDI signals.

The user configures Rooms, i.e. environments, in which he will later improvise. A concert with T^2 usually consists of one 'Room' or a 'Suite' of several rooms, which are called upon in order to improvisationally pass through or move within.

MODULES

Many modules of the same or different types can be part of the same Room. For example, one Listener can be used for the user's input, but another Listener can at the same time listen to the output of a Player in this Room. This way T^2 can also control and adjust its own output. Important modules are: Listener, Player, Modifier and Harmony. These modules are described in more detail in the sections below. Other module types are a Metronome and, of course, input/output modules.

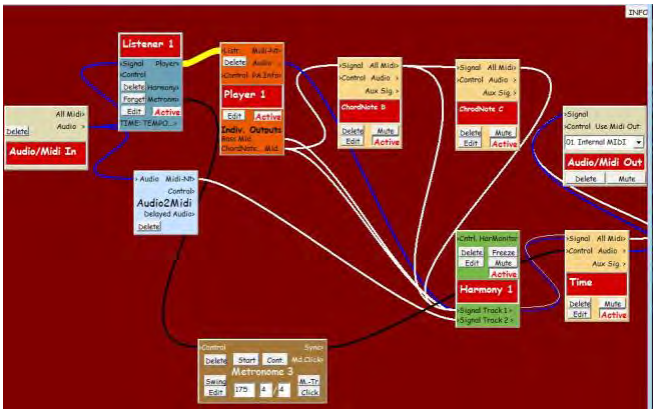

Figure 1. A very simple Tango^2 Room. The video on www.henning-berg.de was produced using this Room

Listener

The Listener remembers the most recent music it received over its signal input – note for note. The user can determine a memory loop for audio and MIDI data,therefore a trail of data for the Evaluation(s) is always available. If audio (converted to MIDI by the Listener's audio to MIDI) and MIDI data arrive in the Listener, both are saved so that the relevant information regarding pitch and volume over the course of every individual note can be accessed at any time. Thus, Tango's audio memory, other than 'normal' sampling memories, knows about the musical content of what is saved. The input is then analyzed with more than 200 parameters. These function on different time scales (three short/long-term memories) and musical topics (e.g. intervals, harmony, tempo, loops, density, surprise etc.). The results of this Evaluation can be scaled and wired for real time control of other parameters during a performance.

Player

The Player uses the Listener's memory to generate Tango's responses. It can operate one or more tracks that play either simultaneously or alternately. With these responses the Player can react and also act in a rather flexible and, if that is desired, surprising way. Each track alone can produce monophonic or polyphonic material, including chords (Figure 1). The Player's output can then be sent to other modules to modify it according to the current needs.

Modifier

The Modifier alters properties of MIDI- or audio-notes (e.g. intervals, volume, legato-quality, quantization). The degree of those changes can be tied to defined conditions and – again – modulated via real time control.

Harmony

The module's purpose is to bend music, which is played by a Player without harmonic objectives, so that the result will sound harmonically organized. This can be done via a Lead sheet (which is common in Jazz), or controlling and modifying the dissonances, as they (happen to) come from a Player. Furthermore there is the possibility to develop individual systems of chords and scales, defining the Harmonic Relevance of each scale note with 'T-shirt sizes' from XS to XL. Harmony can also be controlled by the Listener's harmonic Evaluation or come up with its own variations of given tonalities, e.g. "for harmonic variation use 20% diatonic steps with secondary dominants and only 5% diatonic modulations."

CURRENT AND FURTHER WORK

Tango^2 is a work in progress. As of April 2017 I just finished a Listener function to get a level/percentage of confidence, whether my playing has to do with a certain melody or not – even, when it is played in a rather abstract way with many added or left out notes, different key or tempo or starting in the middle and not from the beginning.

Next will be modules that organize time and rhythm in a more subtle way, something that already existed in the old Tango for the Atari, but had no priority until now for T^2. Also, the GUI needs some work, e.g. groups of modules should work hierarchically to be able to simplify Rooms.

Finally I want to work at modules that play modules rather than music, i.e. work on a higher level. But most of all, I am glad to be able to play concerts with Tango^2 again after many years of developing it.

ACKNOWLEDGMENTS

Thanks to Clarence Barlow, Werner Kracht, George Lewis, Robert Rowe and Christoph Windheuser for many good ideas and great inspiration.

Technology for better improvisation when teaching coding

Matthew Yee-King
Goldsmiths, University of London, UK
m.yee-king@gold.ac.uk

Mark d'Inverno
Goldsmiths, University of London, UK
dinverno@gold.ac.uk

ABSTRACT
Improvisation can be used in teaching coding, when the teacher improvises a program in front of their class. We argue that improvisation allows people teaching coding to build a bridge between themselves and learners. Improvisation has several desirable features: it has a tight action-feedback coupling, it exposes process, it allows mistakes, it provides a shared and novel experience. However, this style of teaching places demands on the software development environment the teacher uses. We derive a set of features for software de velopment environments that address the challenges of the improvisational approach. These features include livecoding capability, realtime audiovisual programming, rapid code sharing, automated code versioning and code replay.

ACM Classification Keywords
Social and professional topics: Computing education; Applied computing: Performing arts

Author Keywords
improvisation; creative coding; pedagogy

INTRODUCTION

Teaching and learning can be seen as a continuum. At one end is the 'sage on the stage', where a well formed body of knowledge is explained by an expert to novices. At the other end is self directed learning, where a learner figures out what they need to know and how to learn it. The former is associated with deductive problem solving, where the solver seeks similarities between the present problem, and previously seen problems and knowledge. The sage provides the knowledge, and the novice has to work out how to apply it. The other extreme is associated with inductive problem solving, where new knowledge is created and applied to the problem at hand, by the learner. There are problems with both these approaches: the deductive extreme does not teach the process through which new knowledge arises and the inductive extreme risks the creation and assimilation of new, but 'incorrect' knowledge, as noted by Hammer [1].

The teaching of coding tends toward the deductive approach, where a series of programming constructs are introduced and demonstrated, such as conditionals, iteration and objects. The learner is expected to pick up these techniques, and to apply them appropriately. Evidence is emerging that the deductive approach is not the best way to approach coding. Yee-King et al. found that high achievers tended to employ an inductive approach [2]. They also found that programming assignments can be designed that encourage all students to engage in more, 'high achiever'-style, inductive programming. In this paper, we propose an improvisational approach to teaching coding, which aims to build a bridge between deductive and inductive approaches. We also propose how we might design improvisational coding interfaces that are suitable for this kind of educationally oriented coding.

IMPROVISATION FOR TEACHING CODING

We ask the reader to imagine a scenario where a teacher is at the front of a class; their laptop screen is projected and the students can see what they are typing, and what the resulting program does. The tutor describes a program they wish to write, for example, a shoot'em up style arcade game, and then proceeds to improvise the code for the game. We draw a parallel here with live, improvised, musical performance such as that seen in a jazz gig. The soloist works their way over the chord progression, 'solving' it as they go. Let us consider which aspects of the improvisational process might contribute to the success of this teaching approach.

Imp1) Action-feedback coupling

Effective improvisation relies upon a tight action-feedback coupling, where an action taken by the improvisor rapidly and tractably results in a response. In musical performance, this might be a response from other band members, from the audience, or simply the experience, by the musician of their own sound. The equivalent in the coding scenario is the translation from code (action) into a running program with output (feedback). This is important so that the students can easily connect the result of the code written to the output generated, as the code is developed. The students are challenged to develop their own expectations about what a given line in the program will do once it is written, then to integrate new knowledge as the result is displayed.

Imp2) Exposing and reviewing the process

Improvisation in musical performance involves real time composition. The process of composition is exposed to the audience. It can be reviewed later, by listening to a recording. The process of writing code is normally, somewhat more obscure.

We are used to interacting with complete programs, for example word processors, where not only do we not have easy, if any access to the code, but also we have very limited, if any, access to the process through which that code was written. By improvising a complete program in front of a class, the tutor exposes the complete process through which a program is written. At the end, the students can review the code, though perhaps not the process. We shall return to that problem later. Exposing process is a key element in bridging from expert to novice – there is no magic (well not much), it is simply a more competent version of their own process. That process is dynamic and it can involve the input of the learner.

Imp3) Making mistakes

It is very likely that the tutor will make mistakes, given the cognitive load of conjuring a solution to a problem, explaining the solution and translating it to code at the same time. Mistakes are an excellent learning opportunity - the tutor can ask the class to attempt to find the problem, the tutor can explain why they made that mistake and then demonstrate how to debug it and solve it. This is another bridging technique – the 'expert' makes mistakes, the novice is engaged in solving them, the novice becomes more like the expert.

Imp4) Shared experience and novelty

Improvisation results in a shared experience, especially considering the above features of rapid feedback, exposed process and mistake making. The students can follow and engage in the improvisation, for example, the questions students ask about the program being written will lead the instructor to cover different technical and creative aspects of programming, just as the jazz players trading fours will respond to each others' riffs, The improvisation also results in novelty – a new program is written, and the process includes student input.

REQUIREMENTS FOR AN IMPROVISATIONAL, CODE TEACHING INTERFACE

Having considered some important aspects of improvisation for teaching coding, we shall now consider how these aspects can be enabled by technology. Consider the scenario mentioned above, where the teacher needs a means to carry out and present their improvisation to and with the class. This technology should take the form of an integrated development environment (IDE), which is a set of tools that enable programs to be written and executed. By considering how to enable the improvisation with technology, it will become clear that a new type of programming environment, or at least, significant new features are required to properly support this process.

Imp1) Action-feedback coupling

Placing a livecoding feature in the code editor, where the program updates automatically as the code is edited, will tighten the feedback loop considerably. Also, the ability to generate interactive graphics and sound, in real time, provides another modality of feedback, more engaging than the typical 'print text to console' approach. Therefore, the technology should allow for real time, audiovisual programming.

Imp2) Exposing process

A simple text editor could be used to edit the code, and the students would then be able to watch the process. However, this can be enhanced if the text editor can be viewed directly on the students? screen, with real time updates. This can be achieved by implementing the editor as a web application with reactive data capabilities, which students can use instantly with only a web browser. Also, how might the student revisit the process for later review? We propose that the technology should automatically store snapshots of the code as it is written, and it should present the student with an interactive timeline of the process.

Imp3) Making mistakes

How might the technology enable the integration of mistake making into the improvisation? As mentioned above, if the students have immediate, real time access to the code as it is edited, they can more easily engage in detecting and fixing problems as they arise. If the students can 'fork' the code, making their own version, they can experiment with solutions on their own code timeline. Also, the automatic code versioning and timeline feature will allow them to review the process through which the mistake was solved.

Imp4) Shared experience and novelty

Building on the idea of a reactive, real time web application, a shared experience naturally emerges. Including ideas of social presence into the design can enhance the idea that the students are in a shared, virtual space together. This might be as simple as showing a list of who is viewing a document, or it could be more sophisticated, showing a dynamic, branching timeline for the program as student add their own contributions to the program during the lesson.

CONCLUSION

We have described how improvisation can be used to bridge the gap between extreme deductive and inductive approaches, and between export and novice programmers in an educational scenario. Based on this, we have proposed a set of features that would allow a software development environment to be used as an effective, improvisational interface for teaching coding.

REFERENCES

1. David Hammer. 1997. Discovery learning and discovery teaching. *Cognition and instruction* 15, 4 (1997), 485–529.
2. Matthew Yee-King, Mick Grierson, and Mark d'Inverno. 2017. STEAM WORKS: Student coders experiment more and experimenters gain higher grades. In *Global Engineering Education Conference (EDUCON), 2017 IEEE*. IEEE, 359–366.

Evaluating Improvisational Interfaces

Toby Gifford
SensiLab, Monash University
Melbourne, Australia
toby.gifford@monash.edu

Shelly Knotts
SensiLab, Monash University
Melbourne, Australia
shelly.knotts@monash.edu

Stefano Kalonaris
SARC, Queens University
Belfast, UK
stefanokalonaris@gmail.com

Jon McCormack
SensiLab, Monash University
Melbourne, Australia
jon.mccormack@monash.edu

ABSTRACT
Improvisational interfaces are human-machine systems for improvising creatively with. To facilitate research in this field, we seek some general approaches to evaluating the quality of a particular interface, and the success of a given improvisation. Drawing on a broad array of improvisational practices across music, visual art and drama, we draw out common threads and discuss the potential for implementing computational models of improvisation that can self-evaluate.

ACM Classification Keywords
Human-Centred Computing: HCI design and evaluation methods

Author Keywords
Improvisation; Computational Creativity; Evaluation

INTRODUCTION
Improvisation is well established mode of creative practice across many different domains: from music, through painting to drama, performance and beyond. Traditionally, improvisation research has mostly considered improvisation between human artists. However, in recent years, interest in improvising with non-human intelligences or systems has grown. We feel that the field has reached a level of maturity that now requires detailed understanding of both the mechanics and psychology of improvisation with non-human partners.

The focus of this paper is in improvisation with computer systems. It is part of a research trajectory in designing, implementing and evaluating *improvisational interfaces* – computer interfaces for improvising in creative arts domains, as outlined in [33]. Ultimately we are interested in realising the notion of machines as *creative collaborators*, and as part of this overall goal we seek to implement improvisational behaviours in machine agents, facilitate improvised exchanges between humans and computers, and create guidelines for designing interfaces that support improvised interaction.

As with any design research methodology, we see *evaluation* as playing a fundamental role [16]. A natural question presents itself: how does one evaluate an improvisational interface? For that matter, how does one evaluate an improvisation?

The design and implementation of machine improvisers and improvisational interfaces more generally is a transdisciplinary endeavor, drawing from several fields, including psychology, computational creativity, and digital musical instrument (DMI) design.

The question of evaluation has been addressed by several authors in computational creativity (for example [26, 39]), in DMI design (for example [25, 4, 11]), and in the intersection of these two fields [43]. A common thread in these discussions is the importance of aesthetic evaluation of creative artefacts. It behoves us then to examine the critical cultures of various creative fields, and the kinds of aesthetic elements that enter into critical evaluations.

EVALUATING IMPROVISATION IN CREATIVE DOMAINS

Creative Improvisation
Many creative domains explicitly involve improvisation, particularly in performance: musical improvisations in various genres such as jazz, experimental, solos in rock; theater improvisation (in for example theater sports); spoken word in poetry slams and hip-hop battles.

Beyond performance, improvisation can play an important role in creative development for many artforms, such as music composition, painting, writing etc. Indeed Sawyer argues that improvisation is the "purest" form of creativity [41], the generative kernel of an iterative process of generation, evaluation and refinement, as depicted in figure 1.

Evaluation as Part of the Creative Process
According to Dewey, "creative activity is our great need, but criticism, self-criticism is the road to its release" [13]. In iterative models of the creative process evaluation plays a key

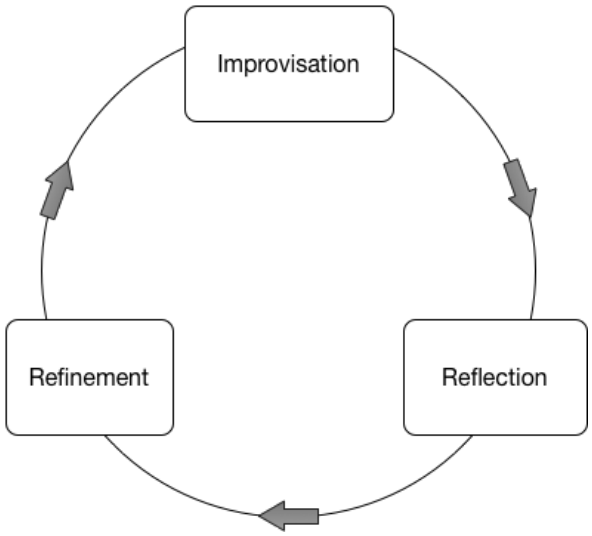

Figure 1. Improvisation in an iterative creative process

role in the iterative cycle, and some researchers argue that the creator is in a constant process of evaluation [35, 22, 41].

How, then, to evaluate an improvisation? As Derek Bailey wryly noted, "any subject is, I suppose, fair game for the academic, but it does seem that if there is anything which is singularly inappropriate for treatment in this way it must be free improvisation" [3]. Yet improvisation in any creative culture has 'rules' be they overt or covert.

Anything Goes?

A recurring theme in commentary from practitioners and theorists alike is countering the myth that in improvisation "anything goes". For example:

> Improvisation is *not*, as it is sometimes perceived, completely free, or "doing whatever you feel like," although it strives for new and less restricted avenues of expression than the current set of rules can accommodate – a process often involving intuitive and even rash decisions. It requires skill and training, it can be learned, and it can fail horribly, precisely *because* there are tacit rules within the community of improvisers [45].

Similarly, in reference to Jazz improvisation, Sawyer notes that "even the freest improviser plays within a musical tradition, and before you can improvise you have to learn that tradition" [41, p. 337].

No Such Thing as a Mistake?

"If you make a mistake then play the same thing over again and with feeling. Let them know that you meant it" (Louis Armstrong) [32].

Thomas Nunn seeks to dispel myths about absence of mistakes or difficulty in musical improvisation [36], and to, instead, approach improvisation as an expression that needs cultivating and practice, whereby mistakes can be contextualised and retrospectively subverted for incorporation into subsequent improvisations.

This process of recontextualising the unintended lets us consider that in improvised performance, evaluation is performed *prospectively* rather than retrospectively. The important question is not 'how good was that?', but rather 'what can I do with that?'.

Cultures of Critique

One important methodological consideration that has received much support from researchers is the use of subjective aesthetic assessment by 'domain insiders' as an instrument for evaluating improvisation [43, 37, 30]. This parallels similar arguments in creativity assessment theory [1, 17].

In designing improvisational interfaces, it may be that the designer themselves play the role of expert judge. For example, in the context of designing DMIs, Jordá and Mealla comment:

> [Just as] much research in HCI culminates in lists of guidelines and/or principles for design (and/or evaluation of design) based on research or practical experience relating to how people learn and work, it comes as no surprise that the first tentative NIME design frameworks have been mostly proposed by experienced digital luthiers [25].

Ben Swift argues a similar line, suggesting that the term 'criticism' may be more apt than 'evaluation' in the context of DMI design:

> The concept of criticism, rather than evaluation, also provides an explicit scope for expert judgments. As Bardzell notes, expert judgments happen all the time in design, whether implicitly or explicitly. This has always been true for the design of musical instruments . . . we must not be afraid to take the same methodological stance in the design of [digital musical instruments] [44].

Embracing criticism as evaluation promises rich returns, since all creative domains have some culture of critique, and corresponding critical values. The difficulty of making explicit those values remains however. As Eisenberg and Thompson note, "the evaluation of improvised music is particularly mysterious" [17].

ASPECTS OF IMPROVISATION

Caines & Heble (2015) have compiled a collection of essays from improvisation practitioners and theorists of many persuasions. Reflective of the broad themes arising from the diverse contributions to the topic, they curatorially categorise these essays into seven aspects of improvisation, which also provides a point of departure for discussing the broad church of improvisation – listening, trust/risk, flow, dissonance, responsibility, liveness, surprise. We have chosen to variously conflate, modify, and extend their categories into the list: listening & liveness; trust, risk & responsibility; flow and rupture; and novelty & surprise.

Trust, Risk and Responsibility

From Caines and Heble's more extensive list we've grouped together Trust, Risk and Responsibility as relating to the performer's interaction with the social and physical constraints of improvisation. Improvisation fundamentally involves live exchange with audience, collaborators and interfaces, and the dynamics and boundaries of those interactions define the creative output. Trust, Risk and Responsibility are some of the parameters that define that interaction, and enable the performer to perform at all.

Trust, particularly, can be viewed as a heuristic decision making construct which helps the parties involved in deliberating under conditions of uncertainty, which could potentially lead to inaction or paralysis. In this sense, trust is a construct/mechanism by which social complexity is reduced and actions made possible. In a musical improvisation context, trust could express the confidence one player has with respect to appropriate musical response and responsibility on behalf of the other player(s). Trust thus expresses the willingness of one party to rely and accept the actions of another party, often relinquishing control over the outcome.

Trust operates across many timescales of interaction: a virtuoso musician spends years developing an in depth knowledge of their instrument developing technique and muscle memory. The improviser expects in performance that they will get a predictable output from a given input to the interface, and their trust in this allows them to concentrate on the creative side of the performance and to take calculated risks in performance. In a collaborative context, we can refer to "yes, and . . . " rule of theater improvisation [14]. Performers trust that their collaborators will respond in a positive and creatively synergistic way to their contributions to a performance. One of the challenges of building computational improvisation partners has been developing this trust in given inputs generating predictable, but creative outputs.

These multiple layers of trust are important in allowing an improviser to take calculated risks in performance. We'll explore risk here through a case-study from current practice in improvised computational arts. Risk is clearly exemplified in the growing field of Live Coding - a practice where performers write and modify their algorithms on stage to produce sound, visuals and other creative outputs. Live coders not only risk creative embarrassment, but risky programmatic moves may produce catastrophic system crashes or punctuate the performance with lengthy debugging processes. Live coders project their performance interfaces, giving the audience insight into their creative and computational processes, but also access to their computational failures. Culturally however this risk-taking is seen as part of the art-form. Anecdotally audience members have commented that crashing is 'just what happens' in live coding, and in more intimate settings 'community bug fixing' can become part of the performance narrative. Whereas in other computational arts rigorous testing is required to ensure system stability, live coders trust in their algorithmic knowledge to code their way out of creatively sticky corners and technical proficiency to avoid and solve errors. Arguably, risk is vital to generating novel creative outputs in performance - performers may combine algorithms in new ways and push the envelope of known outputs through complex algorithmic design.

Strongly related to trust and risk, responsibility may refer to the social responsibility to communicate with and respond to collaborators. Social interplay is key to successful collaboration, and improvisers continuously change roles during a performance, switching between leading, following, and taking part in duets, trios, etc. The dynamic nature of role assignment brings with it the necessity for performers to constantly by conscious of providing space for other performers but also taking the lead and contributing new ideas when creatively necessary. In computational creative domains we might also add to the social responsibility of the performer, the social and cultural responsibility of the technology designer [31, 27]. Beyond designing stable and reliable tools, the designer should also be conscious of how the affordances of their interfaces may impact on the socio-cultural factors of improvisation, and performers with appropriate levels of creative agency in performance.

Liveness and Listening

Improvisation is about the here and now – it is essentially situated in character [23]. The ability to respond to changing or unexpected circumstances – this is the hallmark of improvisation, but it is also the hallmark also of 'liveness'.

What is it about live performance that thrills so, audience and performer alike? This has been the subject of much discussion in many art-forms, particularly music and theatre/film. In the context of digital media, prominent theses are those of Auslander [2] who, channelling Walter Benjiman, argues that live performance nows serves the role of 'authenticating' recorded works, and Dixon [15] who finds performance flourishing in technologically mediated artforms. Common threads include the demonstration of virtuosity and the concommitant necessity for risk-of-failure.

A common refrain amongst jazz musicians is the importance of being responsive when improvising [33, 5]. To improvise one must be aware of context, and synergistically interacting with other improvisers.

> A central element of the phenomenon of peak or elevated performance in jazz improvisation is the placement in time on the part of the player. By this I do not mean rhythmic time, but rather a quite special sense, or 'trifocal' awareness, of precisely (i) where one is in the heat of the creative moment, but with a simultaneous comprehension of (ii) where one in the context of the ensemble has come from and (iii) where one (and the ensemble) is going [23].

What of solo improvisers? Is this same sensitivity to context then important? In this case it becomes the audience with whom the improviser's social contract is formed. In this sense the solo improviser is akin to any live performer. The live performer must construct the performance anew each time – conscious of the mood of the crowd, the feel of the place. The DJ, though often maligned by traditionalists as not really

performing, is fundamentally focused on this key facet of liveness.

Some of the authors' previous experiments in machine improvisation have suggested that an awareness of, and responsiveness to, the actions of a human improvising partner, are among the most important traits of a machine improviser – in terms of fostering a sense of creative partnership [10]. Furthermore, in terms of information architecture, aesthetic evaluation performed by the machine improviser should be applied to the entire ensemble's output as a whole (rather than, for example, simply the machine's own output) [20, 21].

Flow and Rupture

An important phenomenological facet of improvisation is the experience of getting 'in the zone'. Hagberg notes that Jazz improvisation creates in performers an "unmistakable sense of the inimitable uniqueness and non-arbitrariness of a performance in the zone – a state that has remained, despite its undeniable importance and special value in musical performance, interestingly easier to recognize than to fully understand and describe" [23].

Cognitive Flow

Reports, such as above, of a sense of focus and timelessness in performance have been defined by Czikzentmihalyi as cognitive Flow. This unhindered creative state can generally only be achieved when a performer is able to interact with their tools unobstructed by technical or physical inefficiencies, and crucially when various factors align to allow the performer to enter the correct cognitive state to be creative.

Flow as ease of interaction

Related to the experience of cognitive flow is the ability of the performer to have smooth and fluid interactions with an creative interface. Musical instruments have been refined over centuries such that the ergonomics of the interface allow a skilled musician to interact effortlessly with it to produce target sounds.

In designing new interfaces those that allow immediate, straight forward and unobstructed interaction with a creative output may have more appeal than those that are difficult to play or use. However, as McDermott et al. [34] argue, 'ease' in the typical HCI sense is not necessarily desirable in an improvisational interface. As discussed above, in order to obtain Flow, there should be a balance of skill and challenge – an interface that presents no challenges also does not afford demonstration of virtuosity in performance.

Flow and Rupture as generative principles

Another description of flow in relation to improvisation is given by Rose from the point of view of an observer. In breakdancing Rose describes the smooth transitions made by the performer from one point to the next."Flow stands for the possibility while composing that each next move, whether abrupt or continuous, could present itself without hesitation or interruption." [18].

Rose formulates this observance of flow as a generative device whereby new material is produced out of the performer finding a new state which can be smoothly transitioned to from the last state. Rose's generative formulation relies on the idea of rupture as the antithesis to flow. Rupture is a point of stopping, changing direction or friction in the movement.

In relation to our current context on continuous evaluation, we could see rupture as reflective points where a performer breaks out of flow - generative, cognitive or interactive - to evaluate the performance and current context and decide on their next move. In this sense, aspects of friction and resistance may be just as important as flow in improvisation contexts as building opportunities for rupture into a system, interface or performance forces a performer to pause and reflect.

Detecting and Evaluating flow

The presence of flow is usually determined through various subjective observational factors evaluated after a creative activity such as experiencing timelessness, clarity and high concentration levels [24]. However, detectable physiological markers such as heart rate and breath depth have also been found to correlate with subjective observance of flow [12].

Performer feedback alongside physiological sensing could be useful in determining relative levels and occurrence rates of flow and indicating rupture frequency. This may allow an interface designer to determine the right level of friction into an interface and to determine the creative value of system resistance. Though physiological evaluation is likely only practical on a performance time scale [37], a diary of flow observance could apply across all time scales of interaction with an interface.

Novelty and Surprise

In simple terms, novelty is something either you, the world or the universe haven't experienced before. More formally, novelty has been well studied in the creativity literature (see e.g. [7, 42]) where discussions about the relevancy of appropriate novelty play out (the qualitative difference between a statistical outlier and something perceived as genuinely surprising and new). More recently the concept of "novelty search" [29] has found application in evolutionary search. In this technique – inspired by the non-objectivity of real biological evolution the algorithm searches for regions of the search space not visited previously, deliberately avoiding the traditional fitness optimisation of standard evolutionary search. While mainly suited to deceptive problems, the idea has been further extended using "minimal criteria" [28] to reduce the overall size of the viable behaviour space and to try to minimise success of unsuccessful phenotypes.

Novelty and surprise are intrinsically linked to expectation. 'When a listener (or performer as listener) hears what they expect, there is low complexity, and when they hear something unexpected, there is a higher complexity.'[8, p. 4] Therefore, one could say that low complexity, or low entropy in information theoretical terms, violates surprise. The local importance of a particular musical event might vanish over time but, if subsequent material seem to relate to it, then its relevance will increase, prompting its inclusion in some schema which might help the forming of expectation regarding future events.

Some computer simulations of musical expectations have been used for computational music analysis [38] and for generative

music production [9]. More broadly, computational evaluation of the balance of novelty and surprise has been part of several machine improvisation systems [6, 40, 19].

CONCLUSION

In order to progress research into improvisational interfaces, a set of guidelines, critical values, and methodologies for evaluating both interfaces and improvisations is desirable. Developing these is complicated by the diverse array of creative practices that involve improvisation, the variety of contexts in which such interfaces may be used, and the difficulty in formalising criteria for what makes a successful improvisation in any field. Nevertheless, every creative practice has a culture of criticism, and several common threads can be drawn from these that suggest general properties of successful improvisations, and in turn suggest methodologies for evaluating the efficacy of a given interface in facilitating improvisation.

Notions of *trust, risk and responsibility* are endemic in improvisation. One important aspect of implementing a computational collaborator is the handing over of responsibility for some part of the music production to the computer, with concomitant relinquishment of some degree of control. This requires trusting the machine to do something appropriate and interesting. Any useful evaluation of an improvisational interface will likely need to address this dimension, in terms of successful risk taking on the part of the machine, as well as the facilitation of successful risk taking on the part of the human.

Achievement of *Flow* is an important indicator of successful improvisations. Thus evaluating the attainment of flow by the performer is one approach to evaluating the interface with which they perform. Binary assessments of whether inflow or not are readily self-reported by improvisers after the fact. More fine-grained measures of level-of-flow may also be gleaned from a variety of physiological markers.

A sense of *immediacy* is another key factor to evaluate in improvisational interfaces. The inherent temporality of improvisation implies that improvisational interfaces need to support immediate expression. This in turn implies the need for a rich input interface, whether by virtue of fluid gestural interfaces, or by leveraging rich physical systems such as acoustics or image processing.

Finally, the field of improvisation is broad and crosses many creative domains and creative cultures. As such no single evaluative framework is likely to fit all circumstances. Yet one methodological maxim seems to be universally applicable - ultimately improvisational interfaces should be evaluated through use, according to the quality of the interaction they engender, and their utility in furthering and facilitating creativity over many timescales, from a moment of performance to a life of learning.

ACKNOWLEDGMENTS

This research is supported by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (project DP160100166). The views expressed herein are those of the authors and are not necessarily those of the Australian Government or Australian Research Council.

REFERENCES

1. Teresa M Amabile. 1996. Creativity in context. Westview Press, Boulder, Colo.
2. Philip Auslander. 1999. Liveness : performance in a mediatized culture. Routledge, London ; New York.
3. Derek Bailey. 1993. Improvisation: Its nature and practice in music. Da Capo Press.
4. Jeronimo Barbosa, Joseph Malloch, Marcelo Wanderley, and Stephane Huot. 2015. What does "Evaluation" mean for the NIME community. In NIME 15. Baton-Rouge, LA, 156–161.
5. Paul F Berliner. 2009. Thinking in jazz: The infinite art of improvisation. University of Chicago Press.
6. Peter Beyls. 1988. Introducing Oscar. In Proceedings of the International Computer Music Conference. ICMA.
7. Margaret A Boden. 2004. The creative mind: Myths and mechanisms. Psychology Press.
8. D. Borgo and J. Goguen. 2005. Rivers of Consciousness: the nonlinear dynamics of free jazz. In Jazz Research Year Book, L. Fisher (Ed.).
9. Andrew R Brown, Toby Gifford, and Robert Davidson. 2015. Techniques for generative melodies inspired by music cognition. Computer Music Journal 39, 1 (2015), 11–26.
10. Andrew R. Brown, Toby Gifford, and Bradley Voltz. 2016. Stimulating Creative Partnerships in Human-Agent Musical Interaction. Comput. Entertain. 14, 2, Article 5 (Jan. 2016), 5:1–5:17 pages.
11. Dom Brown, Chris Nash, and Tom Mitchell. 2017. A User Experience Review of Music Interaction Evaluations. In NIME 17. Copenhagen, 370–375.
12. Örjan de Manzano, Töres Theorell, László Harmat, and Fredrik Ullén. 2010. The psychophysiology of flow during piano playing. Emotion 10, 3 (2010), 301–311.
13. John Dewey. 1930. Construction and criticism. Vol. 1. Columbia University Press.
14. Dan Diggles. 2004. Improv for Actors. Allworth Press, New York.
15. Steve Dixon. 2007. Digital performance: a history of new media in theater, dance, performance art, and installation. MIT press.
16. Peter Downton. 2003. Design Research.
17. Jacob Eisenberg and William Forde Thompson. 2003. A Matter of Taste: Evaluating Improvised Music. Creativity Research Journal 15, 2-3 (Jul 2003), 287–296.
18. Susan Leigh Foster. 2015. Improvised Flow: Opening Statements. In The Improvisation Studies Reader: Spontaneous Acts. Routledge, New York, NY, USA.
19. Toby Gifford. Appropriate and Complementary Rhythmic Improvisation in an Interactive Music System. Springer London, London, 271–286.

20. Toby Gifford and Andrew Robert Brown. 2011. Beyond Reflexivity: Mediating between imitative and intelligent action in an interactive music system. In 25th BCS Conference on Human-Computer Interaction.
21. Toby Gifford and Andrew R Brown. 2013. Cybernetic configurations: characteristics of interactivity in the digital arts. (2013).
22. J. P. Guilford. 1967. The nature of human intelligence. McGraw-Hill, New York.
23. Garry L. Hagberg. 2017. Jazz Improvisation and Peak Performance: Playing in the zone. In Culture, Identity and Intense Performativity, Tim Jordan, Brigid McClure, and Kath Woodward (Eds.). Taylor & Francis, Chapter 10.
24. Joel M Hektner, Jennifer A Schmidt, and Mihaly Csikszentmihalyi. 2007. Experience sampling method: Measuring the quality of everyday life. Sage.
25. Sergi Jordà and Sebastián Mealla. 2014. A methodological framework for teaching, evaluating and informing NIME design with a focus on expressiveness and mapping. In NIME, Vol. 14. 233–238.
26. Anna Jordanous. 2012. A standardised procedure for evaluating creative systems: Computational creativity evaluation based on what it is to be creative. Cognitive Computation 4, 3 (2012), 246–279.
27. Shelly Knotts. 2015. Changing Music's Constitution. Leonardo Music Journal 25 (2015), 47–52.
28. Joel Lehman and Kenneth O Stanley. 2010. Revising the evolutionary computation abstraction: minimal criteria novelty search. In Proceedings of the 12th annual conference on Genetic and Evolutionary Computation. ACM, 103–110.
29. Joel Lehman and Kenneth O Stanley. 2011. Novelty search and the problem with objectives. Genetic Programming Theory and Practice IX (2011), 37–56.
30. Adam Linson, Chris Dobbyn, and Robin Laney. 2012. Critical issues in evaluating freely improvising interactive music systems. In International Conference on Computational Creativity. 145.
31. Thor Magnusson. 2009. Epistemic Tools: the Phenomenology of Digital Musical Instruments. Ph.D. Dissertation. University of Sussex.
32. Wynton Marsalis. 2013. Twitter Post. twitter.com/wyntonmarsalis/status/325303713809981440. (2013).
33. Jon McCormack and Mark d'Inverno. 2016. Designing improvisational interfaces. In Title: Proceedings of the 7th Computational Creativity Conference (ICCC 2016). Universite Pierre et Marie Curie.
34. James McDermott, Toby Gifford, Anders Bouwer, and Mark Wagy. 2013. Should music interaction be easy? In Music and human-computer interaction. Springer, 29–47.
35. Michael D Mumford, Wayne A Baughman, and Christopher E Sager. 2003. Picking the right material: Cognitive processing skills and their role in creative thought. (2003).
36. Tomas E Nunn. 1998. Wisdom of the impulse: On the nature of musical free improvisation. Thomas E. Nunn.
37. Sile O'Modhrain. 2011. A Framework for the Evaluation of Digital Musical Instruments. Computer Music Journal 35, 1 (2011), 28–42.
38. Marcus T. Pearce and Geraint A. Wiggins. 2012. Auditory Expectation: The Information Dynamics of Music Perception and Cognition. Topics in Cognitive Science 4, 4 (2012), 625–652.
39. Graeme Ritchie. 2007. Some Empirical Criteria for Attributing Creativity to a Computer Program. Mind & Machines 17 (2007), 67–99.
40. Robert Rowe. 1992. Interactive music systems: machine listening and composing. MIT press.
41. R Keith Sawyer. 2011. Explaining creativity: The science of human innovation. Oxford University Press.
42. Robert J Sternberg and Todd I Lubart. 1999. The concept of creativity: Prospects and paradigms. Handbook of creativity 1 (1999), 3–15.
43. Dan Stowell, Andrew Robertson, Nick Bryan-Kinns, and Mark D Plumbley. 2009. Evaluation of live human–computer music-making: Quantitative and qualitative approaches. International Journal of Human-Computer Studies 67, 11 (2009), 960–975.
44. Ben Swift. 2013. Chasing a feeling: Experience in computer supported jamming. In Music and Human-Computer Interaction. Springer, 85–99.
45. Rob Wallace. 2015. Improvisation and the Making of American Literary Modernism. In The Improvisation Studies Reader: Spontaneous Acts. Routledge.

Computational Method for Periodic MIDI Sequences and Percussive Improvisation

Clayton Kardas
Rochester, NY, USA
clayton.kardas@gmail.com

Ming-Lun Lee
Rochester, NY, USA
minglunlee@rochester.edu

ABSTRACT
This paper discusses a computational method to create periodic MIDI sequences which allows improvisation in real-time using percussive pads from a MIDI controller, creating a multi-voiced MIDI output. The sequence's relation to modal music theory limits the improvisation parameters. Depending on the mode (Lydian, Ionian, Mixolydian, Dorian, Aeolian, Phrygian, Locrian) and sequence's pitch at a specific instance, users are given four of the individual pitches of an arpeggio with which to improvise over. These pitches are based on the root note of the arpeggio being the current pitch associated with the periodic MIDI sequence being played. These parameters prevent users from creating inharmonic sounds and help plan out musical passages consisting of chord sequences for real-time performances.

ACM Classification Keywords
Human-computer interaction; Real-time systems

Author Keywords
Percussion; MIDI; Electronic;

INTRODUCTION
This research project was made in the hopes of giving drummers the opportunity to incorporate tonal music theory into modern day drum sets; allowing for drummers to become a more integral part of the creative process and sonic accompaniment for modern day ensembles of musicians. Using Pure Data, a visual programming language for processing audiovisual media, a patch was created (synonymous with a programming script) that allows for the periodic cycling of arpeggios and chords of MIDI numbers that drummers can play in real-time using an electronic MIDI controller with percussive pads. The cycled groups of pitches are based on the roots of a metronome-like MIDI sequence and both are outputted through different MIDI channels. Thus, giving percussionists multiple voices to control in real-time.

METHODS

Periodic Sequences
The MIDI sequenced voice represents the metronome for the chord sequence. Instead of having the metronome play a uniform pitch with each tick, the model opts for the ticks to be replaced by pitches that represent the current root note. Each tick iterates a variable b by one and is initialized with 0 (shown by the variable b++). When b is put into a modulo function, which outputs the remainder after division, with variable mb,

$$y_1 = (b{+}{+}) \,\%\, \text{mb} \tag{1}$$

the output y can appear periodic. This can be seen in Figures 1 and 2 where mb is 3 and 5, respectively. This computational method uses an array L filled with MIDI numbers and can be cycled through periodically with the variables r and mc used in the same equation as Equation 1. Equation 2 iterates through the array elements in L with variable r initialized at zero and iterated by 1 each time y from Equation 1 equaled zero.

$$y_2(y_1) = \begin{cases} (r{+}{+}) \,\%\, \text{mc}, & \text{if } y_1 = 0 \\ r \,\%\, \text{mc}, & \text{otherwise} \end{cases} \tag{2}$$

The number of elements that the user wants to use in L decides the integer for mc. For example if mc equaled three, L was an array with five elements, and r was initialized at zero, then the periodic sequence y2 would only cycle through the first three elements of L. These elements represent rows of L, which is a multidimensional array. The first column of each row can represent the possible root numbers within a melody where each row represents a musical section and can hold several other variables such as mb. If each row's mb variable is uniform then the output would look like Figures 1 and 2 if mb was 3 or 5, respectively.

If L is only using two rows (mc = 2) and has mb as 3 for the first row and 5 for the second, and cycled through those two roots when r and b were initialized at 0, then the output would look like Figure 3. This output looks to be random and disjointed, but the beginning of the next period can be seen when b equals 15. A slightly different periodic structure can be seen when r is initialized with 1, shown in Figure 4.

This shows that they have different transients, but ultimately create the same periodic sequence. Initializing a sequence with a phase shift can create transients. This effect can be seen when r is initialized at 0, 5 is the mb for the first root number, and 3 is with the second root number. Overall, this periodic sequence represents the metronome separated into

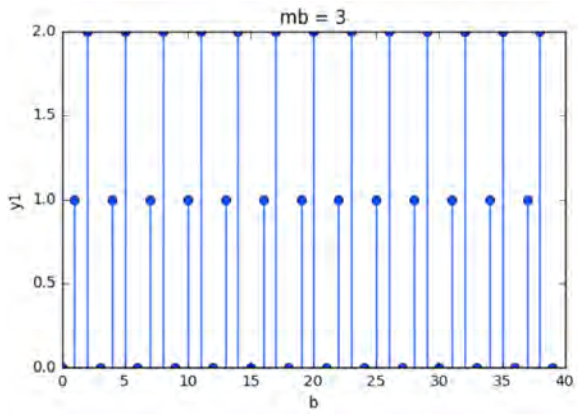

Figure 1. Output for Equation 1 when 0 <= b <= 40 and mb = 3.

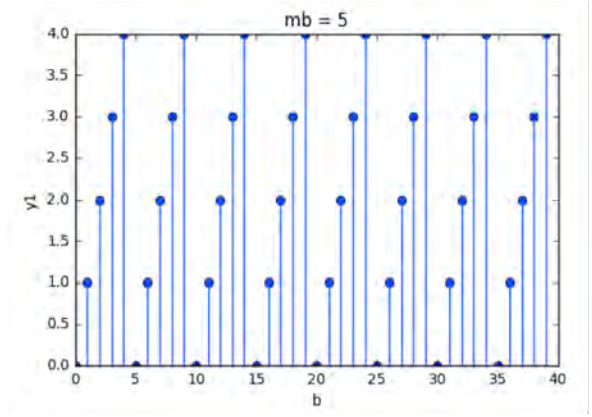

Figure 2. Output for Equation 1 when 0 <= b <= 40 and mb = 5.

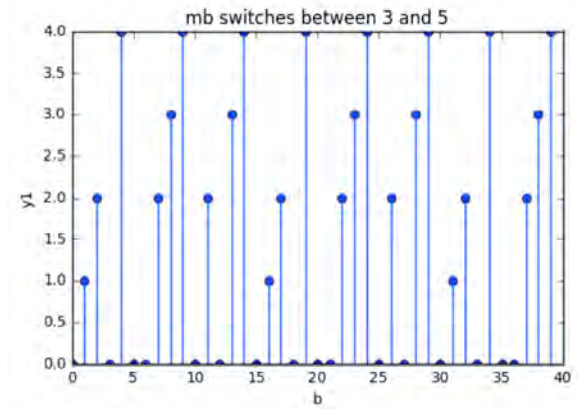

Figure 3. Output for Equation 1 with mb = 3 or 5 when 0 <= b <= 40 and switches whenever y1 is equal to 0.

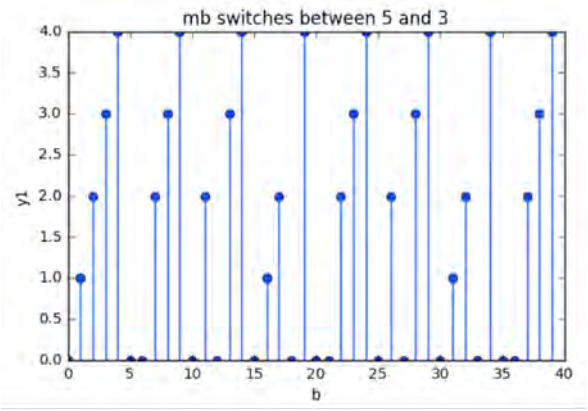

Figure 4. Output for Equation 1 with mb = 5 or 3 when 0 <= b <= 40 and switches whenever y1 is equal to 0.

discrete sections when 0 is the first number. The patch, which performs these computations, is shown in Figure 5.

Melody Subpatches

The melody subpatch (a patch within a patch) in the original PD patch serves as a structure that holds array L, integer mc, and another periodic iteration called y3 seen in Equation 3

$$y_3 = (tn{+}{+})\,\%\,\mathrm{mn} \tag{3}$$

to classify which melody subpatch of multiple melody subpatches are playing at a specific time. This is controlled with another operation like equation 1 but with a variable *tn* that is similar to *r* and *b*, and integer *mn* which indicates the number of melody subpatches available during operation. Each melody patch can hold a unique L array with the output of the melody subpatch's root numbers controlled by a unique mc integer. The variable y_3 is used in each melody subpatch as part of a conditional operation. The user can have N amount of melody subpatches, which are numbered 0 through N-1 and use the Boolean output of equation 4.

$$y_3 == N \tag{4}$$

Equation 4 is to toggle which melody subpatch is in operation. This is shown with sections of the PD GUI shown in Figure 6 and Figure 7, which give an overview of two melody subpatches connected to equation 3 and the details within a melody subpatch. As seen in Figure 7, the conditional output toggles the current integer of the variable mc and the root numbers of array L to be played during that melody subpatch?s periodic cycle. During each periodic section that begins with y = 0, the root note of that section serves as variable m, which is the root note at a particular point in time. This becomes important with the modal music theory involved with this model.

Mode Subpatch with Percussive Pads

The variable *m* serves as the current root number and is considered when finding out the appropriate arpeggios. The other two inputs are the key-note number which is any number from 21 through 32 which represent the lowest 12 chromatic MIDI numbers (*A*, *A*♯ / *B*♭, *B*, *C*, *C*♯ / *D*♭, *D*, *D*♯ / *E*♭, *E*, *F*, *F*♯ /

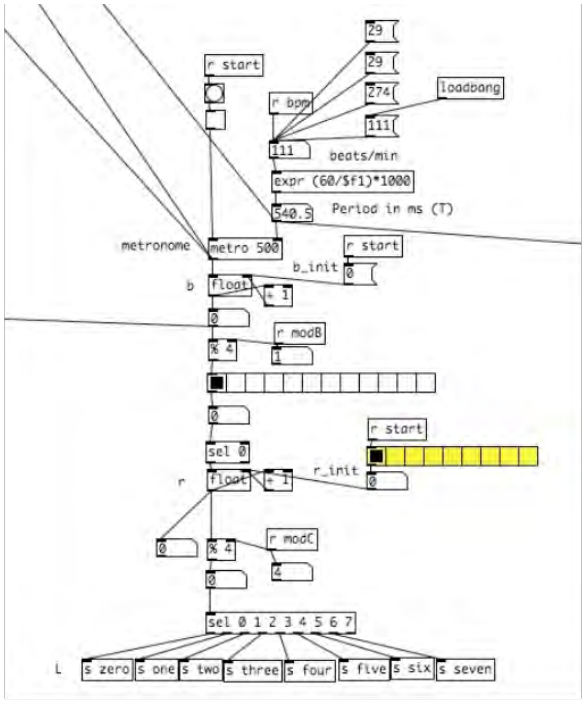

Figure 5. Screenshot of the Pure Data configuration for creating a periodic sequence.

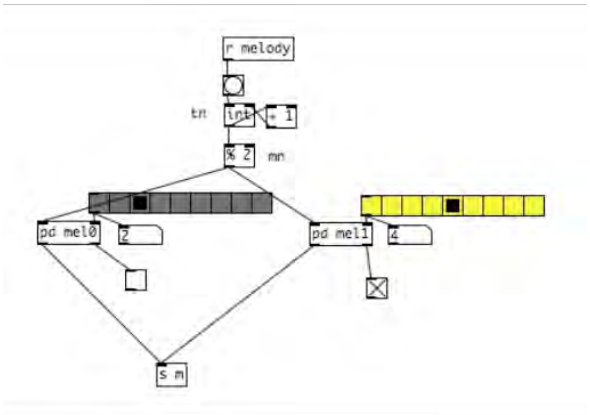

Figure 6. Screenshot of the Pure Data configuration of two melody subpatches.

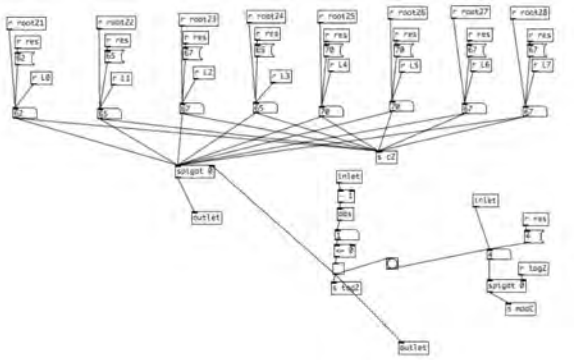

Figure 7. A Screenshot of the layout within the melody1 patch.

$G\flat$, G, $G\sharp$ / $A\flat$), and the mode selector which uses numbers 0 through 6 to represent the seven modes mentioned earlier, respectively. An arpeggio can be formed based on these parameters. According to standard music theory (e.g. [3]) the seven different modes each have a scale degree that is flatter than the previous mode, and triad chords are created with a third and fifth scale degree from a root scale degree. Having an octave from the root pitch available with the triad chord creates an arpeggio. Once these arpeggio note numbers are available after computation from the mode subpatch, they can be assigned to individual percussive pads. Therefore, the user is able to play notes from arpeggios based on the variable m, the assigned key pitch, and mode.

Faraday Array

The Faraday array sequencer was found on the website of Andrew Faraday [1] while searching for ways to play pitches other than the variable m during melodic sections using appropriate intervals from major or minor scales. This would allow for the creation of melodic passages by using an equation similar to equation 1.

$$y_4 = (b{+}{+}) \,\%\, \mathrm{ma} \tag{5}$$

Equation 5 is has mb switched to ma, where ma is another number associated with periodicity and the output y4 is used to choose an element of the Faraday array. Each element in the array holds an interval that serves as the scale degree of a major or minor scale and adds this interval to the root note. The major or minor scale can be determined by using the subtraction of the third scale degree (3 for minor, 4 for major) from the number 3 to equate to a Boolean definition (0 or 1). This then uses a case structure to select 0 for minor or 1 for major. It also unlocks the potential of creating unique periodic patterns that have not been fully explored. They will be discussed in the results section.

MIDI Controller and Setup

This computational method for music can be used with any MIDI controller that has percussive pads. For this paper I used a V49 Alesis MIDI keyboard, a Yamaha DTX-400K electronic drum set, and a Pyle PTED01 electronic drum set. There have been many setups with the combinations of these hardware instruments and some of those will be discussed in the general

Figure 8. 4x2 Percussive Pad Layout for the V49 MIDI Controller..

discussion. For simplicity's sake, an explanation with the V49's operation will be presented here. The V49 possesses a 4x2 set of percussive pads. These pads are used and labeled as seen on Figure 8.

The bottom four pads allow the user to play the arpeggio's root, third scale degree, fifth scale degree, and octave pitches by learning the pad?s original MIDI number and activating the arpeggio MIDI numbers upon striking of the appropriate pad at a particular time. The channel for the arpeggio pitches is separate from the sequenced pitches. The sequenced pitches can be turned on and off with the 'On/off' button and the melody subpatches iterate *tn* by one after hitting the 'Melody' button which switches to the next successive melody subpatch. This channel can be isolated by choosing channel 10 of the 16 available channels associated with the controller since 10 is the default for percussive instruments. A third channel can be used to play the instrument of the MIDI controller (usually channel 1) whether it is keyboard or MIDI guitar. This gives the user an opportunity to create multiple layered voices with their own unique style.

RESULTS

Prior to improvisation, the user must build certain parameters, which can be modeled using this method: melodyN([LN(root = rn, ma = ♯, mb = ♯)], mc = ♯, FN) where N can be any integer between 0 and infinity, ♯ represents any integer between 1 and infinity, rn represents the root note for the particular L array row, LN represents the unique L array as a whole, and FN represents the Faraday array in the melodyN subpatch. There are still many limitations within the software, but also many examples that show it can be a viable candidate for helping musicians and lay-people create music and exercise theoretical music knowledge. The first example will show the voices for the sequencer, percussive pads, and keyboard where it periodically cycles through melody0([L0(root = 60, ma = 4, mb = 4), L1(root = 64, ma = 4, mb = 4)], mc = 2, F0 = [0,3,5,8,3,0,5,4]) with the mode settings in C Ionian (also called C Major). The integers in the Faraday array elements indicate the scale degree of the interval to add to the root note. The roots 60 and 64 are C3 and E3, respectively. Also, note that the figures below are screenshots of a piano roll from Ableton Live that is 'folded' to only show the pitches activated during a recording session. Figure 9a shows one period of the sequencer playing the C Major arpeggio and then the E minor arpeggio because it only cycles through the first four elements of the Faraday array for each L in melody0. Figures

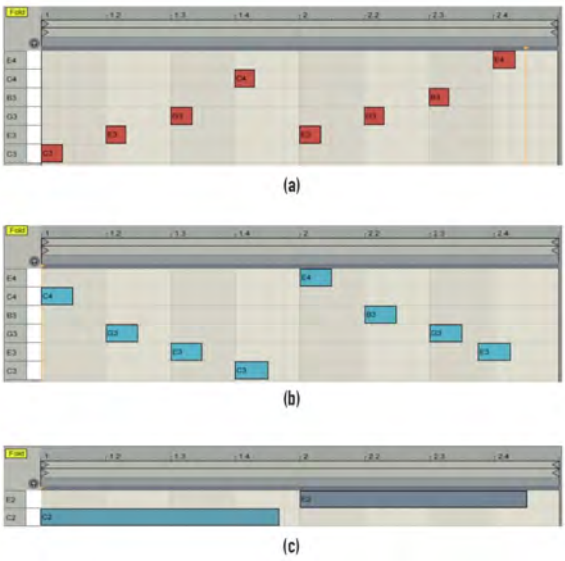

Figure 9. Quantized recording of the (a) MIDI sequence only playing the first four elements of the Faraday array, (b) percussive pad pitches, (c) keyboard pitches

9b and 9c show the same arpeggios descending from the octave and the notes C2 and E2 being played on the keyboard keys, respectively. Figure 9 is an example that shows what happens when all the ma and mb variables are uniform in an L array. The second example shows the MIDI sequence when ma of L1 is changed to 1. Figure 10 shows the sequencer producing a C Major arpeggio in the first section L0, but repeats the root note in the second section L1. Figure 11 shows the sequence when ma of L1 is changed to 8 and is able to play the full Faraday array. This is because when row 1 on the L array is activated and ma is changed to 8, the output of y4 is [4,5,6,7] which are the last four elements of the Faraday array, continuing where y4 from row 0 of the L array left off. Figure 12 shows switching between two melody subpatches. The subpatch melody0 will remain the same as the examples above and the other subpatch will be: melody1([L0(root = 62, ma = 1, mb = 2), L1(root = 64, ma = 8, mb = 8)], mc = 2, F1 = [0,12,3,0,3,0,3,0]) . This example is shown for two reasons; the first reason is that melody1 has a configuration that allows for elements to be skipped with the Faraday array. A twelfth scale degree is never played because the elements 0, 2, 3, 4, 5, 6, and 7 are the only elements to be activated in the array. This means that element 1 can be any arbitrary number and the user can aim for activating specific sections of the array. The second reason is for showing the transients that can occur during switching and the ability to produce multiple periodic outputs that is dependent on timing of the switch.

Figure 12 shows a continuous sequence that is 22 bars long and divided into four quarter notes for each bar. For an example of notation, bar 5 at quarter note 2 is represented by 5.2. A black box can be seen around the sequenced note when the percussive pad for Melody was hit to switch to the next melody. The first being at 2.4 on figure 12a where it is switching from melody1 to melody0, and then back to melody1 at 4.4 with no

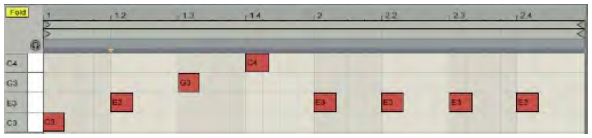

Figure 10. Quantized recording of the MIDI sequence with the first section playing first four elements of the Faraday array and the second section playing the first element of the Faraday array.

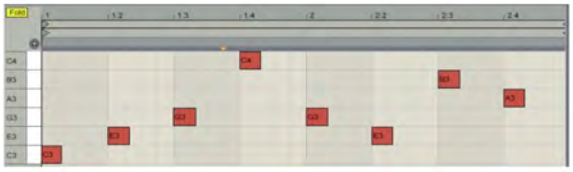

Figure 11. Quantized recording of the MIDI sequence with the first section playing the first four elements of the Faraday array and the second section playing the last four elements of the Faraday array.

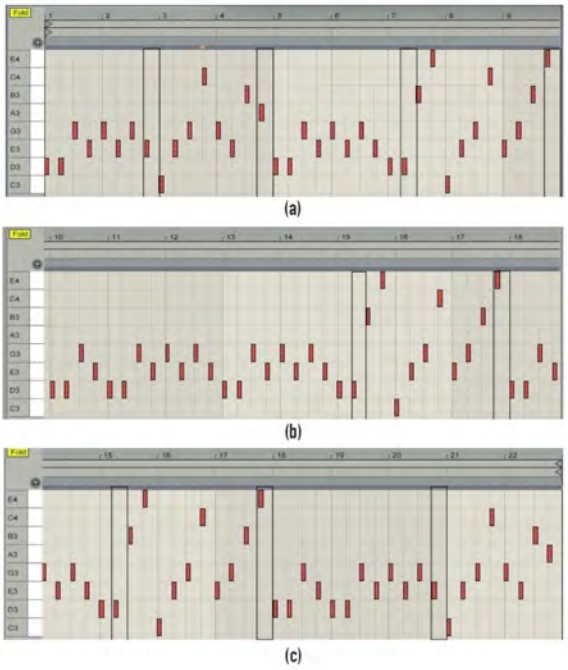

Figure 12. Quantized sequence of 22 bars with black boxes surrounding the instances when a Melody switch was made.

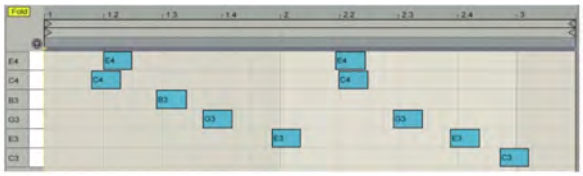

Figure 13. Unquantized recording of improvisational percussive hits with offbeats.

transients between either switches. At 7.2 a melody switch is made, but melody0 does play the pattern from Figure 11, but from Figure 9a starting at 2.3. The mechanisms that involve the two differ ent melody0 periods are unclear. One possibility is that the periodic sequence switches within a periodic phase the periodicity associated with Figure 9a, thus being able to only cycle through the first four elements of the periodic array. Another possible reason could be that L0 and L1 from melody0 have integer multiples of ma and mb with L1 from melody1. There might be other possibilities, but regardless of the reason, it switches back to melody1 at 9.4. Throughout the next bar on Figure 12b there is a transient in bar 10, which then stabilizes into its normal periodic form at 11.1. There is then another switch to melody0 at 15.2, which shows that switching at that point can consistently cause melody0 to play the same as what follows 7.2. After switching again at 17.4, shown in figure 12b and figure 12c, there is a similar transient that follows like in bar 10. It is then switched once more at 20.4 to play a period of the full Faraday array. Figure 12 in general shows examples of transients between melodic switches and how melody subpatches can be shown to play differently depending on certain conditions. These conditions seemed to be based in modular algebra and applied math, which would be a good research topic following this report.

GENERAL DISCUSSION

Described so far is a computational method for generating metronome-like melodic sequences and being able to improvise over the sequence based on the pitch being played at a particular time. The concept of combining multiple MIDI controllers, use with music therapy, music education, the foresight of possible areas of improvement, and a general evaluation of the software?s performance is discussed next.

Combining MIDI Controllers

Using the MIDI controllers listed above in the Methods section, multiple configurations were created with which to make music. One of the first configurations was with the DTX400 electronic drums and the V49. The DTX400 has four circular percussive pads, three cymbal-modeled percussive pads, two foot pedals, and a module for connecting all the pads together which has a USB cable for its MIDI output. With this model, one can replace the pads associated with the toms for the chord triad, leaving the octave out and not completing the arpeggio. The On/Off button and Melody button remain on the percussive pads of the V49 shown in figure 8. This combination allows the user to play the original electronic drums such as the snare, bass, ride, crash, and hi hat cymbals, and the ability to activate MIDI numbers based on the

MIDI sequence. Another possible configuration combines the DTX400 and Pyle PTED01, which has seven percussive pads and two foot controllers. With those extra percussive pads and the creation of a subpatch that outputs pentatonic scales (five pitches per octave) rather than triads, users can have access to two octaves worth of major or minor pentatonic scales based on the variable m and the key. The first three MIDI numbers can be mapped to the toms of the DTX400 and the following seven pitches can be mapped out to the percussive pads of the PTED01. The On/Off and Melody buttons can be mapped to the two foot pedals, which allows a more convenient work flow for drummers using this method.

Music Therapy

Percussion is a large part of music therapy for patients of all kinds [2]. With the combination of percussion and music theory that is based on predetermined parameters, patients may be able to create music that was not possible for them to make before. Patients with Parkinson?s disease could have the opportunity to create melodic music more easily by having the metronome-like sequence guide their gait restrictions [4]. This approach can also pave the pathway to incorporating more music education into music therapy.

MUSIC EDUCATION

Music education in general could benefit from this computational method by allowing users to have real-time interaction with chords and chord sequences. In its current form, the model can help musicians map out sections of songs that follow a format that is similar to pop, rock, hip-hop, and blues genres. This method has the potential to extend to sequencing between complex chords and scales on the percussive pads. There is also potential for the creation of specialized MIDI controllers with percussive pads to fit certain performance requirements. A possible software update could be varying the duration of the pitches played from the MIDI sequence and percussive pads instead of being uniform throughout a session. Creating intervals between the variable m and the arpeggios created could lead to music creation more similar to voice leading to where the multiple pitches played are distinctly individualized instead of being harmonically related. The input for the Faraday array could also be improved since it currently only accepts positive numbers; optimizing it for negative scale degrees could widen the possible range of the output. Because the current configuration has the MIDI sequence and the improvised pitch output to different channels, this allows for users to separate them into different audio tracks in a DAW. So, when playing with an ensemble, the ensemble, through ear monitors, could only hear the MIDI sequence and the improvised pitches would be complementary with the rest of the ensemble?s voices. Conversely, the MIDI sequence could be played along with the ensemble or the sequence could be turned on or off throughout a piece. There are many different ways this instrument could be used either alone or with a group of musicians.

Possible Areas of Improvement

A possibility would be to recreate the program in another scripting language and consider the current version a prototype rather than a final product. The creation of an intuitive user interface is also needed. Regardless, this type of computational method for creating musical sections with which to improvise opens doors for new types of musical creation and algorithms that can further optimize this method as a promising new electronic instrument.

General Evaluation

The amount of potential for sounds and patterns, which can be produced, seems nearly infinite. Because the software can adapt to any MIDI controller with percussive pads, controllers with a large array of pads can host large sets of possible scales, modes, or arpeggios for each root note in the sequence. The amount of notes mapped and root notes used in a melody is reciprocal with the amount of diverse sounds that can be made. With so many options, the possibility of improvising for an extended period of time is high.

REFERENCES

1. Andrew Faraday. 2013. Sequencing and wavetables, part 1: Arrays in Pure Data. (May 2013). `www.andrewfaraday.com/2013/05/sequencing-and-wavetables-part-1-arrays.html`
2. Andrew Knight and Bill Matney. 2012. Music therapy pedagogy: Teaching functional percussion skills. *Music Therapy Perspectives* 30, 1 (2012), 83–88.
3. Mark Levine. 2011. *The jazz theory book*. " O'Reilly Media, Inc.".
4. Matthew WM Rodger and Cathy M Craig. 2016. Beyond the metronome: Auditory events and music may afford more than just interval durations as gait cues in Parkinson's disease. *Frontiers in neuroscience* 10 (2016).