pOM: Linking Pen Gestures to Computer-Aided Composition Processes

Jérémie Garcia[1],[3]
[1]INRIA, UPSud, CNRS (LRI)
F-91405 Orsay, France
jgarcia@ircam.fr

Philippe Leroux[2]
[2]Schulich School of Music,
McGill University &CIRMMT,
Montreal, Canada
philippe.leroux@mcgill.ca

Jean Bresson[3]
[3]STMS Lab: IRCAM-CNRS-UPMC
1, place Igor Stravinsky,
F-75004 Paris, France
bresson@ircam.fr

ABSTRACT

Illuminated manuscripts of medieval music contain rich decorations in addition to handwritten neumatic notation. Our project with composer Philippe Leroux investigates the use of such handwritten symbols during the composition of his piece *Quid sit musicus*. We introduce *pOM*, an interactive paper application and a library for the OpenMusic computer-aided composition environment which links pen gestures over an old manuscript to compositional processes. The paper interface analyzes the stroke while writing and transmits several features to reactive programs in OpenMusic. *pOM* allows the composer to define his own functions and get direct musical feedback from pen interactions.

1. INTRODUCTION

Contemporary composers use computer-aided composition (CAC) environments to create and run programs that combine algorithmic processing with musical material or notation. CAC tools feature advanced computational possibilities but are usually more limited in terms of interaction, which can prevent composers to express their musical intentions. As Eaglestone and Ford [1] argue, these systems mostly focus on enabling the technical implementation of musical processes, rather than actually supporting composers' creativity. In general, composers interact with CAC tools using a mouse and keyboard, when they could also benefit from using other input and output devices [2].

The New Interface for Musical Expression (NIME) community studies new hardware and software for musical expression but generally emphasizes performance over composition [3]. Usual approaches use gesture recognition [4] or mapping techniques [5] to feed real-time sound synthesis processes with data coming from sensors.

The present work concerns the design of an interactive system that combine CAC and interactive paper interfaces to support composers' creative process. Interactive paper technology captures hand-written gestures on paper and transmit them to the computer. In previous works, we explored several possibilities using this technology to incorporate pen interactions into compositional processes.

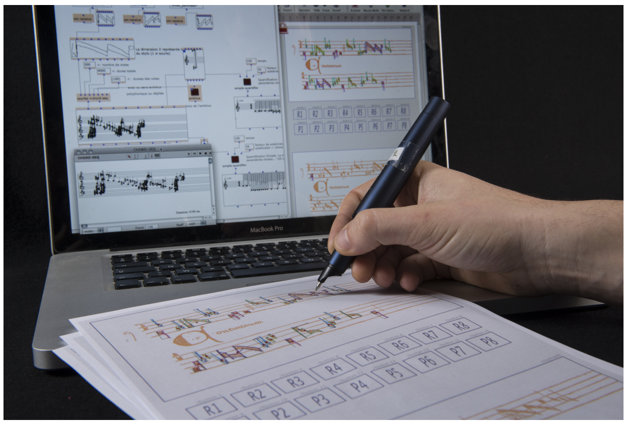


Figure 1: *pOM* in action. Pen events on the paper interface trigger musical processes in *OpenMusic* which and generate direct audio-visual feedback. Photo H. Raguet ©Inria.

For example, the *InkSplorer* system allowed composers to experiment with curves drawn on paper to control computer-based algorithms, and enabled interactive and simultaneous work on both media [6].

This paper presents *pOM* (literally *pen-OpenMusic*): a project focusing on the integration of interactive paper in the *OpenMusic* environment, and in the realm of computer-aided composition in general. It details our work with Philippe Leroux during the composition of *Quid sit musicus*.[1] An interactive paper interface captures and interacts with drawn symbols over an old manuscript. The composer used this interface along with an *OpenMusic* library to generate compositional material. Figure 1 shows the general framework of the project. On the computer, a server application receives and handles pen interactions from the paper interface, and an *OpenMusic* patch transforms the transmitted data to control compositional processes.

Section 2 presents the musical motivation and background of this work from the composer's point of view. We then describe the main technological components of the project, namely, a pen-and-paper server system (Section 3) and a prototype reactive framework developed in OpenMusic (Section 4). We detail this framework and its use by Philippe Leroux in Section 5. In Section 6, we discuss how *pOM* helped the composer to explore and refine his musical processes using pen interactions, and present perspectives for the project.

[1] The piece *Quid sit musicus* was commissioned by Ircam and premiered at the Manifeste festival in June 2014.

2. MOTIVATION: CALLIGRAPHIC GESTURE AND COMPOSITION

Philippe Leroux has long been interested in the ideas of writing and gestures in his composition [7]. For example, his pieces *VOI(REX)* (2002), or *Extended Apocalypsis* (2006/2011) use written letters, words or shapes as control data for CAC processes [8]. In these earlier pieces, the composer used a mouse and/or a graphics tablet to input gestures.

An important missing feature for the composer with these interfaces was the availability of *traces* from the written gestures, like the ink a pen would leave on a sheet of paper. When writing on paper, composers can analyze, react and correct their work depending on these traces of their own gesture, but graphics tablet and other usual input devices do not provide such tangible visual memory.

For the composition of *Quid sit musicus*, Philippe Leroux wanted to reinterpret the calligraphic gestures of a manuscript from Guillaume de Machaut, a composer from the fourteenth century. The manuscript, similarly to other medieval "illuminated manuscripts" [9], contains rich decorations in addition to handwritten neumatic notation. Figure 2 shows the principal excerpt of the manuscript of the piece *Ma fin est mon commencement* by Machaut that we have used both as a use case and as a compositional input in this work.

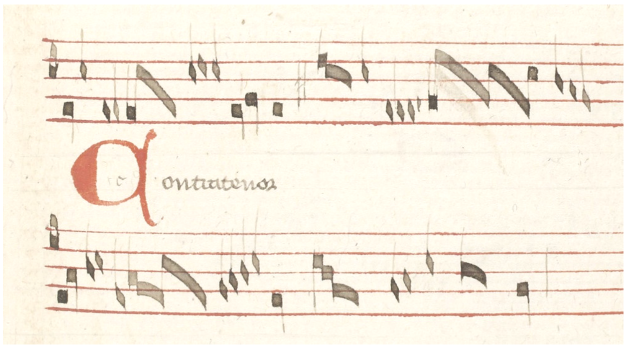


Figure 2: Original illuminated manuscript.

In his piece *Je brûle, dit-elle un jour, à un camarade* (1991), Philippe Leroux already used neumatic notation to write the final score. However, he did not use CAC during the composition. In the present project, he was interested in giving new meanings to the calligraphic forms in the medieval manuscript by using several of their graphical and dynamic properties in algorithmic or electro-acoustic processes. Image-based scanning and recognition techniques could enable the conversion of the neumatic shapes into musical information [10], but they could never recover information generated from the gestural act of actually writing the symbols. For this reason, the composer decided to trace, himself, some excerpts of the manuscript with a digital pen in order to generate and receive information from accurate calligraphic gestures.

3. INTERACTIVE PAPER INTERFACE

Interactive paper technology consists in tracking the movements of a pen over a piece of paper and sending them to a computer for processing. A popular approach to implement such system is the Anoto technology, [2] which uses a digital pen embedding a tiny video camera that detects the precise location of each gesture with respect to a barely visible dot-grid printed on the paper following unique and localized patterns.

Previous works explored how this technology could support the work of contemporary music composers. The *Musink* project [11] showed that they could intuitively use it to develop personal notations on paper over time. The annotations over printed scores were recognized and mapped to online functions. However, this recognition occurred only occasionally once the pen strokes were uploaded to the computer, not while the user was actually writing.

More recent digital pens such as the ADP301 can stream events in real-time via a Bluetooth connection. It is now possible to interact directly with the computer from the paper interface. *InkSplorer* [6] or *PaperTonnetz* [12] are projects that use this kind of pen to let composers interact in real-time with printed paper interfaces. In *PaperTonnetz*, a printed Tonnetz [3] is used as input for the generation of melodic or harmonic material. Pitches are played while the user is drawing, when the pen enters or leaved regions of the Tonnetz. Drawn paths can then be retrieved and heard as chords or melodies thanks to additional pen-based pointing interactions.

For the present work we developed and used a Java application (Figure 3) as a server managing connections between the paper interface and the other applications involved in compositional processes (in particular, the *OpenMusic* environment). This server application can create, lay out and control interactive components on a virtual page, and connect these components to other applications via the OSC protocol (Open Sound Control [13]). It allocates the Anoto pattern to the pages and components before printing them, and interprets the pen events at run-time, sending adequate data to the connected applications. The "digital ink" (a memory of pen gestures) can be stored and reused in later sessions. Figure 3 presents the paper interface we created with Philippe Leroux. It shows two instances of a *Machaut* component containing a background image. This component detects intersecting strokes to recognize shapes while the user is drawing, and computes features from these shapes. The instance on the top already contains written input data.

With our server, each written stroke also becomes an "interactive component", allowing further pen interactions. For example, the user can "click" over a previous stroke with the pen to trigger actions, as one might do with a mouse on a standard GUI button. During these pen interactions on paper, the server sends OSC messages containing the extracted features. A description of this interface and its interaction with *OpenMusic* is given in Section 5.

[2] www.anoto.com

[3] A Tonnetz is a tone network which lays out the pitch as a graph according to interval relationships.

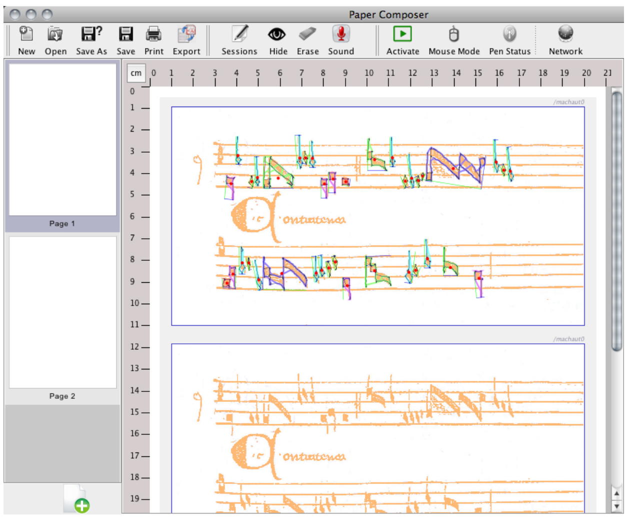


Figure 3: Graphical user interface to create, print and execute paper applications. The virtual page on the center displays the interface, digital strokes and visual feedback.

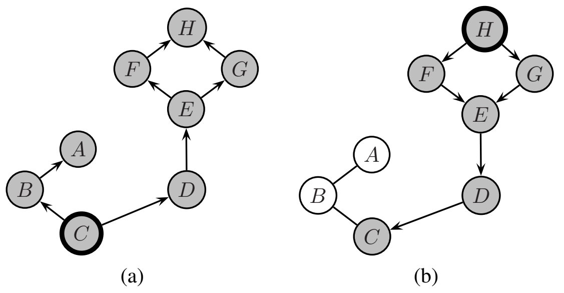


Figure 4: Computation of a visual program represented as a graph. (a) **Demand-driven model**: the value of box C is requested by the user and triggers the evaluation of the upstream boxes in the graph. (b) **Reactive model**: the box H has changed and propagates an update in downstream boxes whose value can be influenced by this change.

4. REACTIVE PROGRAMS IN OPENMUSIC

The *OpenMusic* computer-aided composition environment (OM [14]) is graphical and musical extension of the Common Lisp programming language [15]. OM programs are represented as directed acyclic graphs which correspond to functional expressions generating or transforming musical material. They are created by the user connecting boxes (functions, data structures and sub-programs) written either as text (in Lisp) or graphically (as *patches*). Unlike real-time musical systems that react to internal clocks, external stimuli or data streams following a data-driven computation approach [16], computer-aided composition languages like OM execute programs upon user requests in "deferred-time", following a demand-driven strategy (see Figure 4a).

Recently, a hybrid computational model has been proposed that combines this demand-driven approach with reactive event-driven computation within the OM visual programs [17]. In this model, each node in the program graph has a *reactive* status. Reactive nodes transmit update notifications through their output connections upon the occurrence of *events* (see Figure 4b). An *event* in this context can be the modification of a box value, or of a box input value. It can be produced by user actions (e.g. while creating the program), by internal running processes, or by external incoming data (e.g. received via MIDI or OSC messages). Events propagate through reactive branches of the visual program graph, and update (i.e., re-evaluate) downstream-connected boxes.

This reactive model creates new opportunities to support composers' creative activity in *OpenMusic* without requiring them to significantly alter their pre-existing patches in order to make them reactive. It is likely to ease composers' experimental processes: when input data is manually parametrized, the new values automatically propagate to the downstream components of a patch according to the reactive boxes' status. The reactive model also permits OM visual programs to receive data from external processes and applications (in the present case, from the interactive paper server). Incoming received data can then immediately update the connected musical elements in the compositional processes [18].

We developed or extended a number of tools in OM, inspired by equivalent tools in real-time musical environments, that fit this reactive framework. Below are the main new OM boxes we used in this project:

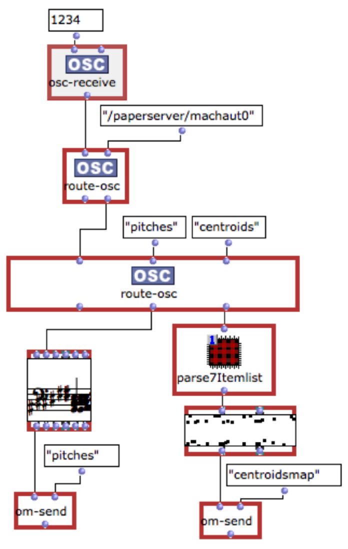
- *osc-receive* runs a server thread listening to an open UDP socket to receive OSC messages. Reception of a message changes the value of the box, triggers an event and propagates a notification in the graph.
- *route-osc* controls the propagation of a notification by testing the address of an incoming OSC message. It also stores in the corresponding outlet the latest messages for each correct tests.
- *om-send* and *om-receive* allow to transmit notifications and data between different locations in the visual program(s), bypassing the graph connectivity.

Figure 5 shows two OM patches that use these tools to process incoming OSC messages. Reactive objects are identified with bold, dark red frames. The patch in Figure 5a contains an *OSC-receive* box that transmits the messages to the *route-osc* objects. The data filtered by *route-osc* are used to build OM objects that are in turn "sent" via *om-send* boxes. In Figure 5b, the *om-receive* boxes receive a notification and update data structures accordingly. [4]

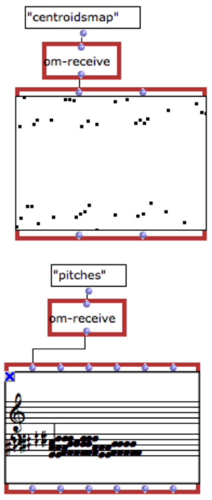
5. *pOM*

The *pOM* system combines an interactive paper interface and reactive programs in OM (Figure 1). The pen server application captures the pen events, recognizes drawn calligraphic forms and sends several of their features to reactive patches in OM. The patches convert incoming data to musical material according to the composer's choices. They also react to specific pen interactions and provide

[4] Note that our "data-flow" description here has mostly an illustrative purpose: in reality only notifications are propagated, and the actual data is computed and passed from one box to another following the demand-driven evaluation of the terminal downstream boxes.



(a)



(b)

Figure 5: Reactive parsing and dispatching of OSC messages in an OM patch. Reactive boxes are displayed with a bold red frame.

direct audio-visual feedback. This section presents how Philippe Leroux used the *pOM* framework during the composition of his piece *Quid sit musicus*.

5.1 Writing on paper and features extraction

The composer begins by tracing on the paper interface to draw his own calligraphic forms over the original ones. The pen-server groups both intersecting and close pen strokes, compared using distance and time thresholds,[5] into single *forms*. Figure 6 shows the composer writing on the paper interface and a detail of the visual feedback of the form detection provided on the server's graphical user interface. For each detected form, this interface displays data for: the strokes, the centroid of the strokes' points, an associated polygon and two main axes. The associated polygon is the convex hull [19] of the form's points. The main axes are two orthogonal segments of the minimal enclosing rectangle of the polygon.

Philippe Leroux wanted to use several features of the calligraphic forms to control his musical processes. The composer defined six categories of forms to match the traditional neumatic forms, e.g. *punctum*, *virga*, *porrectus*. We used simple heuristics with tests on width and heigh ratios of the drawn shapes to classify them according to these categories. The server interface highlights the forms with different colors depending on their category (see Figure 3) In case of recognition errors, the composer can edit the category of a shape with a pop-up menu which appears when he selects a shape on screen with the mouse. Each symbol is also associated to a single point, its centroid, which is used to estimate a pitch in midi-cents precision depending on its position relative to the lines of the staff.

[5] We defined the thresholds from a previous strokes session recorded at the beginning of the project.

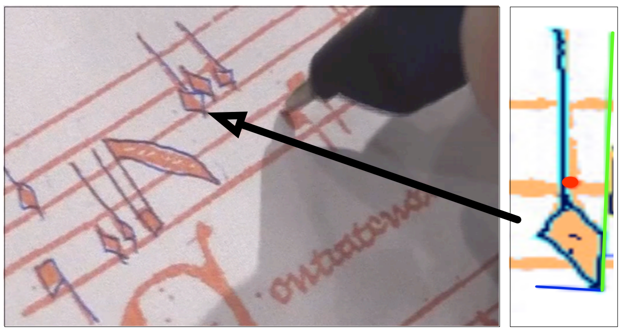


Figure 6: Writing on the paper interface. Left: The composer writing with the digital pen. Right: Visual feedback on the screen including the extracted shape (light-blue), its centroid (red) and main axes (blue and green).

Figure 7 shows different available features, computed by the pen server and received in OM. These features include: the list of strokes, the centroid, the duration, the pitch, the category ("type"), two perimeters, the area, two main axes and thickness envelopes.

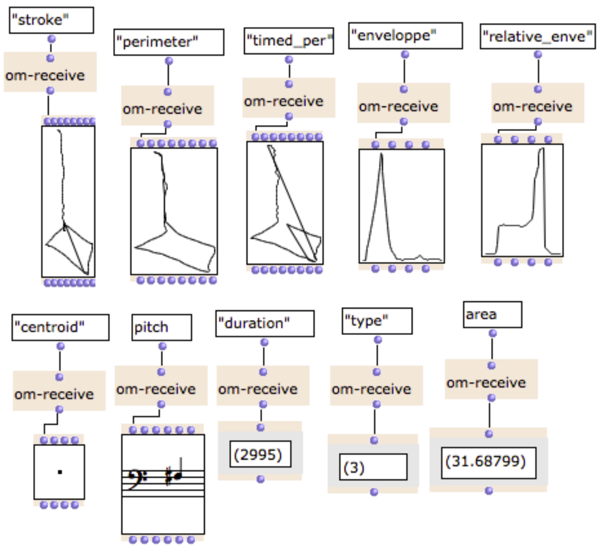


Figure 7: OM patch receiving pen features from the server.

5.2 Pen Interactions

Each time the user writes a new stroke, the pen server updates and transmits features of the current form. It allows the composer to monitor the data coming from the pen while writing, and to visually assess the result of the recognition. The composer can also retrieve the features of a previously drawn shape without modifying it by clicking over an existing form with the pen. This pen interaction uses the drawn strokes and the interior of the associated polygon as an interactive selection area.

5.3 Musical processes in OM

Philippe Leroux designed several processes to create the harmony of the piece, rhythms, melodic and harmonic gestures from the pen data.

The main harmonic scale comes from the detected pitches of the forms in the manuscript (see Figure 5b). For the rhythms, he created *talea*, i.e., rhythmic motives, from proportions related to the classification of the forms according to their category, duration, area or position in the manuscript. Leroux also created additional harmonic and melodic content by transforming the main harmonic scale in various ways, using techniques such as frequency modulation, pitch shifting, distortion or simulation of the Doppler effect. All these processes are controlled with the features extracted from calligraphic forms, and the timing of the resulting musical elements in the piece comes from a classification of these forms.

We adapted the composer's existing OM patches (see [8]) and created new ones to use the new data and take advantage of the reactive framework. Below, we detail three examples of processes involving different features from the calligraphic forms: the strokes points; the speed of the pen movements and the thickness envelopes.

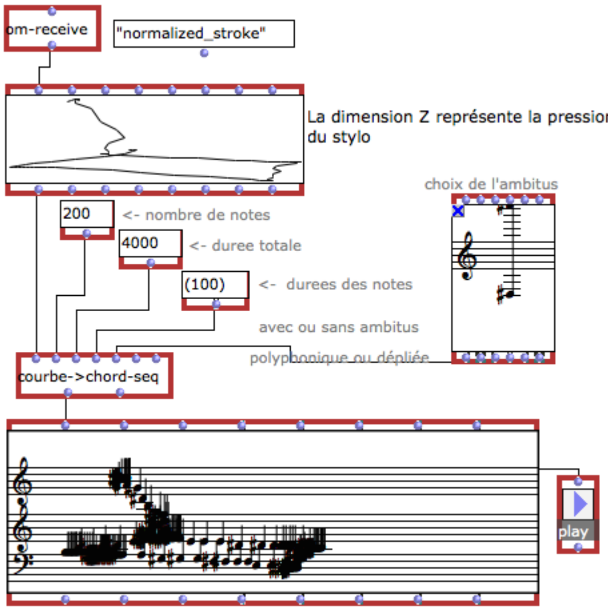


Figure 8: Converting a shape into a polyphonic sequence.

1. The patch in Figure 8 converts the strokes from a form into a sequence of notes by projection of the shape into a score. The patch offers controls for several parameters such as the number of notes, the duration of the sequence, its ambitus and whether it should be polyphonic or "unfolded" in time (see [8]). Another part of the patch, not visible in the figure, filters the sequence with the main harmonic scale pitches.

2. The patch in Figure 9 receives the pen stroke points and computes a speed profile for the whole calligraphic shape. The box *pdoppler* takes a chord and the speed profile to compute a sequence of chords applying a Doppler transformation. Each chord is shifted from a frequency determined by the speed value.

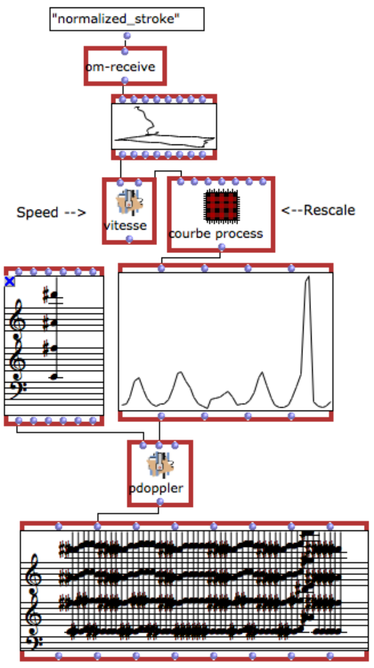


Figure 9: Mapping the pen speed with a Doppler effect transformation.

3. The patch in Figure 10 uses thickness envelopes of a form to distort the frequencies of a chord. The resulting chord-sequence is created from a single repeated chord distorted between the successive pairs of values from the envelopes.

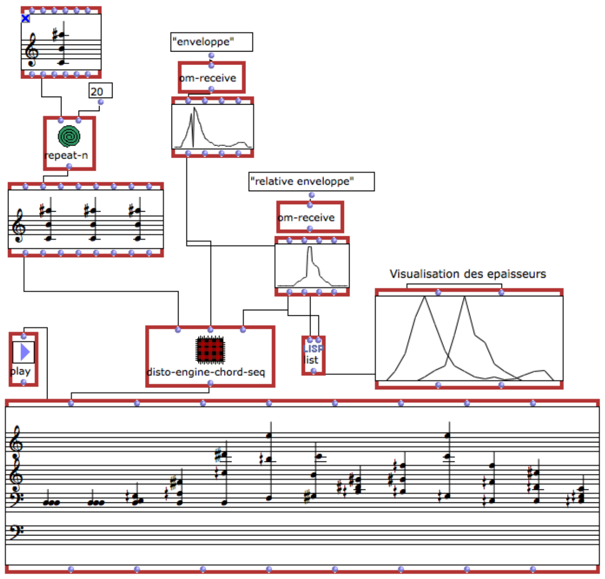


Figure 10: Frequency distortion with thickness envelopes.

In both Figure 8 and Figure 10, a *play* box is at the end of the reactive chain: each time a message is received from the pen server, the sequence is updated and rendered through the OM MIDI player.

6. DISCUSSION

We designed our pen server application and reactive programs in OM to support the input of calligraphic forms and active exploration of compositional processes with a digital pen. Using this framework, Philippe Leroux was able to interactively visualize recognized shapes and features in OM while he was drawing. He could assess and edit the recognition if necessary, either by drawing new strokes or using the graphical user interface. Leroux also frequently used the digital pen as a "pointing" device to select previously drawn shapes and import their features in OM. The composer created reactive patches in which the input data update musical objects, and where the use of *play boxes* (see for instance Figure 8 and Figure 10) establishes a direct causality between drawing gestures and sounding musical results. By mixing pen interactions with his regular use of the computer-aided composition environment, the composer was able to determine which forms were the most interesting. From these forms, he created a set of chords and note sequences that he used in the piece. He explained that he used some of the patches as if he were *"improvising with a piano to search a chord color or a convincing melodic suite."*

In addition to the features available for each calligraphic gesture, the possibility to have data from all the forms inspired the composer to use new processes based on classifications of the forms. In his previous work using a graphics tablet, such data was not available so he did not planned to consider the relationships between different calligraphic gestures as compositional material.

During the composition of his piece, the composer drew several times over different versions of the paper interface to adjust and edit his calligraphic gestures, using the feedback in OM as a reference. Once he was satisfied with a particular realization, he could save it as a new interface including recognized shapes, print it and use it as a new support for interaction. In Figure 3 for instance, the top component of the interface contains recorded forms from a previous session. Figure 11 is the scan of a paper interface used to explore and compare recognized forms. Barely visible dots in the different forms are the traces of pen-pointing interactions.

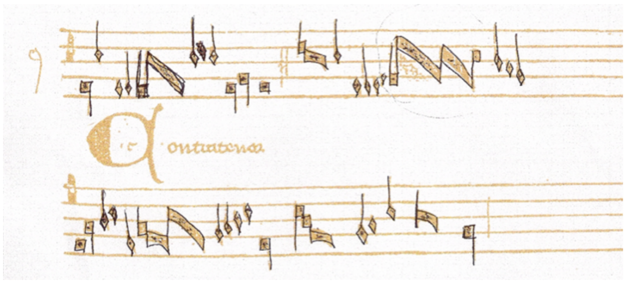


Figure 11: A version of the paper interface used to write and compare the calligraphic forms.

Our initial implementation of the OM library had a background client interpreting incoming OSC messages to update ad-hoc "reactive objects" in OM. As the exploratory nature of the project required frequent modifications of these, we finally designed the current tools bringing visual program reactivity to the front, in order to let the user determine himself the routing of incoming messages, and change the data interpretation with regular visual programming tools. This approach may lead to more complex patches, but it supports quick change in mappings and parametrization. Philippe Leroux continued to take advantage of the reactive properties of the OM boxes after he finished using the paper interface. In particular, he worked on several patches to convert classifications of chords created with the calligraphic forms into control data for sound synthesis.

While the current technological set-up offers promising possibilities for the composer, several aspects could still be improved. First, the composer wished he could to use pressure information but the digital pen does not provide an accurate measure of this parameter while writing. Second, Leroux appreciated interacting with the ink traces left by the pen, but explained that he was missing the ability to erase strokes directly on paper, as he would do with a pencil.

Instead, he needed to print an updated version of the interface before rewriting the incorrect strokes. Previous work proposed methods to support erasing [20] but they require building new ink cartridges for the pen.

Although the work presented here is strongly related to Philipe Leroux's compositional practice, other composers with different approaches could easily design their own OM patches to control different musical processes with pen gestures. The reactive framework can also be used to process OSC messages send by any kind of device, enabling opportunities to integrate additional input and output modalities within the CAC environment [2].

7. CONCLUSION

In this article, we introduced *pOM*, a framework that combines an interactive paper interface with reactive programs in OM. We designed *pOM* in close collaboration with composer Philippe Leroux in order to actualize his idea of controlling musical processes by tracing over calligraphic gestures in an illuminated manuscript. We showed how the composer adapted and used *pOM* during the composition of his piece *Quid sit musicus*.

The reactive framework combined with the pen interactions has proven successful in supporting the exploration of several alternatives as well as refining his musical processes. We believe that this kind of environment could be fertile for creativity as the composer can engage in a partnership with the computer to iteratively improve his patches and programs while exploring input data.

Future work with Philippe Leroux will investigate the control of the spatialization during live performances using this technology. In particular, we are planing to use OM for creating interpolations between trajectories generated with two different forms before sending the result to the spatialization engine. We are also interested in extending and studying *pOM* with other composers.

Acknowledgments

This work has been partly funded by ANR (ANR-13-JS02-0004-01 and ANR-12-CORD-0009). We are grateful to Gilbert Nouno and Dimitri Bouche for their help with the implementation. We also thank Fanis Tsandilas, Carlos Agon, Wendy Mackay and Matthew Schumaker for their help and support.

8. REFERENCES

[1] B. Eaglestone and N. Ford, "Composition systems requirements for creativity: what research methodology," *In Proc. MOSART Workshop*, pp. 7–16, 2001.

[2] J. Garcia, T. Tsandilas, C. Agon, and W. E. Mackay, "Structured observation with polyphony: a multifaceted tool for studying music composition," in *Proceedings of the 2014 conference on Designing interactive systems - DIS '14*. ACM Press, Jun. 2014, pp. 199–208.

[3] T. Magnusson and E. H. Mendieta, "The acoustic, the digital and the body: A survey on musical instruments," in *Proceedings of the 7th international conference on New interfaces for musical expression*. ACM, 2007, pp. 94–99.

[4] J. Françoise, N. Schnell, and F. Bevilacqua, "A Multimodal Probabilistic Model for Gesture-based Control of Sound Synthesis," in *Proceedings of the 21st ACM international conference on Multimedia (MM'13)*, Barcelona, Spain, Octobre 2013.

[5] J. Malloch, S. Sinclair, A. Hollinger, and M. Wanderley, "Input devices and music interaction," in *Musical Robots and Interactive Multimodal Systems*, ser. Springer Tracts in Advanced Robotics, J. Solis and K. Ng, Eds. Springer Berlin Heidelberg, 2011, vol. 74, pp. 67–83.

[6] J. Garcia, T. Tsandilas, C. Agon, and W. Mackay, "Inksplorer: Exploring musical ideas on paper and computer," in *Proceedings of the 2011 International Conference on New Interfaces for Musical Expression NIME2011*, 2011, pp. 361–366.

[7] P. Leroux, "... phraser le monde: continuité, geste et énergie dans l'oeuvre musicale," *Circuit: Musiques contemporaines*, vol. 21, no. 2, pp. 29–48, 2011.

[8] P. Leroux, "The Model of the Model in VOI(REX)," in *The OM Composers' Book - Vol. 2*, J. Bresson, C. Agon, and G. Assayag, Eds. Ircam-Centre Pompidou / Editions Delatour, 2008.

[9] C. D. Hamel, *The British Library Guide to Manuscript Illumination: History and Techniques*. University of Toronto Press, 2001.

[10] A. Fornés, J. Lladós, and G. Sánchez, "Primitive segmentation in old handwritten music scores," in *Graphics Recognition. Ten Years Review and Future Perspectives*. Springer Berlin Heidelberg, 2006, pp. 279–290.

[11] T. Tsandilas, C. Letondal, and W. E. Mackay, "Musink: composing music through augmented drawing," in *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09*, ser. CHI'09. New York, NY, USA: ACM, 2009, pp. 819–828.

[12] L. Bigo, J. Garcia, A. Spicher, and W. E. Mackay, "PaperTonnetz: Music Composition with Interactive Paper Tonnetz," in *Proceedings of the 9th Sound and Music Computing Conference. SMC'12*, Copenhaguen, 2012, pp. 219–225.

[13] M. Wright and A. Freed, "Open Sound Control: A New Protocol for Communicating with Sound Synthesizers," in *Proceedings of the International Computer Music Conference*, ser. ICMC '97. Thessaloniki, Hellas: International Computer Music Association, 1997, pp. 101–104.

[14] G. Assayag, C. Rueda, M. Laurson, C. Agon, and O. Delerue, "Computer Assisted Composition at IRCAM: From PatchWork to OpenMusic," *Computer Music Journal*, vol. 23, no. 3, 1999.

[15] J. Bresson, C. Agon, and G. Assayag, "Visual Lisp/CLOS Programming in OpenMusic," *Higher-Order and Symbolic Computation*, vol. 22, no. 1, 2009.

[16] M. Puckette, "Combining Event and Signal Processing in the MAX Graphical Programming Environment," *Computer Music Journal*, vol. 15, no. 3, 1991.

[17] J. Bresson and J.-L. Giavitto, "A Reactive Extension of the OpenMusic Visual Programming Language," *Journal of Visual Languages and Computing*, vol. 25, no. 4, 2014.

[18] J. Bresson, "Reactive Visual Programs for Computer-Aided Music Composition," in *Proceedings of the the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Melbourne, Australia, 2014.

[19] M. Duckham, L. Kulik, M. Worboys, and A. Galton, "Efficient generation of simple polygons for characterizing the shape of a set of points in the plane," *Pattern Recognition*, vol. 41, no. 10, pp. 3224–3236, Oct. 2008.

[20] S. Olberding and J. Steimle, "Towards Understanding Erasing-based Interactions: Adding Erasing Capabilities to Anoto Pens," *Proceedings of Paper Computing*, 2010.