PaperComposer: Creating Interactive Paper Interfaces for Music Composition

Jérémie Garcia[1,2] *jgarcia@ircam.fr*
Theophanis Tsandilas[1] *fanis@lri.fr*
Carlos Agon[2] *agon@ircam.fr*
Wendy E. Mackay[1] *mackay@lri.fr*

[1]INRIA
Univ Paris-Sud & CRNS (LRI)
F-91405 Orsay, France

[2]STMS Lab: IRCAM-CNRS-UPMC
1 place Igor Stravinsky
F-75004 Paris, France

ABSTRACT
Interactive paper technologies offer new opportunities for supporting the highly individual practices of creative artists, such as contemporary music composers, who express and explore their ideas on both paper and the computer. We introduce *PaperComposer*, a graphical interface builder that allows users to create a personalized interactive paper interface that they can connect to their own computer-based musical data. We also present an API that facilitates the development of interactive paper components for *PaperComposer*. We describe one public demonstration of a novel musical interface designed for children and our collaborations with composers to create two novel interactive music interfaces that reflected their individual composition styles.

Author Keywords
Interactive paper; music composition; creativity; toolkit

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces – Graphical User Interfaces; H.5.5 [Sound and Music Computing]: Methodologies and Techniques.

INTRODUCTION

Music composition has been deeply influenced by the introduction of composition software such as *Max* and *OpenMusic*. These tools allow composers to create new sounds, develop their own musical strategies and explore symbolic notations. However, paper remains fundamental in their creative process [11, 16, 17]. Our goal is to develop interactive systems that support composers' expression and exploration on paper while taking advantage of the computational power of computer-aided composition tools.

Contemporary music composers pose a particularly challenging design problem, because they develop their own unique musical representations on paper and also sophisticated computer programs to express their musical ideas [7, 12, 16]. Any specific solution created for one composer is thus likely to be rejected by other composers. Rather than creating unique interfaces tailored to each composer, we want to develop flexible tools that composers can easily adapt for their own musical needs.

Interactive paper technology creates new possibilities for composers, who will be able to switch between paper and the computer, ideally combining the best of both. The most representative technology for interactive paper is Anoto[1] which combines a small infrared camera embedded close to the tip of the pen, and a tiny, almost invisible, dot pattern printed on paper. The camera detects the pattern and identifies the precise location of the pen on the page while the user is writing.

We are interested in developing a simple toolkit that supports the creation of diverse music-based interactive paper applications. Our goal is to handle different forms of musical data, e.g., musical scores, graphs and personalized musical representations, and link them to a variety of music composition applications, incorporating the standard communication protocols and data structures used by professional composers.

Our previous research introduced the concept of *Paper Substrates* [8]: paper-based composition elements that manage musical data according to specific rules. Composers can create their own personal musical structures and link them to musical data types on the computer. The composer can interact with these musical structures via interactive paper, which can accept and update data from computer composition tools. The first version of *paper substrates* offered a limited set of interactive components and required significant programming to create new ones, limiting the ability of end users to tailor the interface. The next step is to create tools that facilitate the creation and deployment of new *paper substrates*, so that composers can adapt them for their own musical needs.

We introduce *PaperComposer*, an interactive interface builder for creating and customizing *paper substrates*. We describe the API that supports the creation of paper components and present specific examples of interactive paper composition applications created with *PaperComposer*. We discuss the current implementation and conclude with suggested directions for future research.

IHM'14, October 28–31, 2014, Villeneuve d'Ascq, France. Copyright 2014 ACM 978-1-4503-2935-4/14/10...$15.00
http://dx.doi.org/10.1145/2670444.2670450

[1] http://*www.anoto.com*

RELATED WORK

Researchers have developed a variety of interactive paper solutions for diverse domains. For example, PaperLink [2] lets users create hyperlinks from physical documents to electronic content. Strip'TIC [14] is designed to support air traffic controllers by augmenting paper flight strips with pen-based and tangible interactions, and includes visual feedback overlaid on the strips.

Musically oriented interactive paper applications pose special challenges, since composers often sketch on paper to express personal or ill-formed musical ideas that cannot be represented easily with traditional musical representations [11, 12, 16, 21]. For example, Musink [22] helps composers define their own annotations with respect to musical scores and link them to computer-based functions. InkSplorer [7] lets composers draw hand-written curves to control computer programs and PaperTonnetz [3] lets composers explore melodies and chords within hexagonal-shaped sets of tone networks. *Paper substrates* [8] support pen input and other forms of tangible interaction with the paper interface, such as moving, associating and layering paper components to create higher-level musical entities. *Paper substrates* handle musical data from computer-aided composition applications and provide visual structures to support data entry, data editing, and interaction with the pen.

Creating advanced interactive paper applications from scratch requires significant programming effort. Commercially available software development kits offer basic functionality for creating interactive paper applications with Anoto technology, but focus primarily on document management, especially note-taking, editing printed documents and filling out forms. Anoto offers an Adobe Acrobat plugin that allows end users to create, print and interact with forms that content text fields and check boxes. Livescribe offers somewhat more functionality via an Eclipse plugin that lets developers place arbitrary shapes onto a paper interface and link them via pen events. However, this tool requires additional Java code to interpret pen events and create standalone pen applications, and the user must install both the pen application and the paper interface on the pen in order to use them.

Researchers have explored several frameworks and toolkits that speed development and support creation of more complex interactive paper applications. For example, iPaper [18] is a framework that supports rapid development and deployment of interactive paper applications. Programmers define active areas or specialized components and link them to specific services such as sending an email with handwritten content. Paper Toolkit [24] is a generic event-based architecture and toolkit for developing paper user interfaces. Letras [13] assists the deployment of paper applications where interaction is distributed across different input and output devices. Each of these programming environments is designed to help experience programmers to develop highly specialized applications. Unfortunately, they are not suitable for composers or other end users, because they require general programming skills. Note, however, that composers *do* learn to use certain specialized music composition programs, such as *Max* and *OpenMusic* [1]. These allow users to visualize and assemble higher-level musical primitives into networks of numerical values, controllers, graphs, as well as handle classical music notation. Composers also use the Open Sound Control (OSC) protocol [23] to exchange musical data across applications.

Paper Composer is designed for composers who are familiar with these standard, higher-level music composition programs. They can create personalized interactive paper interfaces that control these higher-level musical primitives, which enables them to explore ideas informally on paper and then fluidly move to controlling them on the computer. The goal is to extend existing computer-based composition tools to support the early and intermediate phases of creative exploration, with smooth transitions between simple marks on paper to complex interactive functions on the computer.

PAPERCOMPOSER: AN INTERFACE BUILDER FOR INTERACTIVE PAPER

Our goal is to enable composers, as end users, to create, manage and use their own interactive paper interfaces without necessarily having to program. We decided to build upon the concept of *paper substrates* [8] since they allow end users to create personal structures on paper for specialized musical content which is then interpretable by music-programming software. Previous studies [7, 8, 16] show that composers appreciate the new possibilities offered by interactive paper but also require support for their current composition environments. They also require functionality that simplifies stroke recognition and supports archiving and reuse.

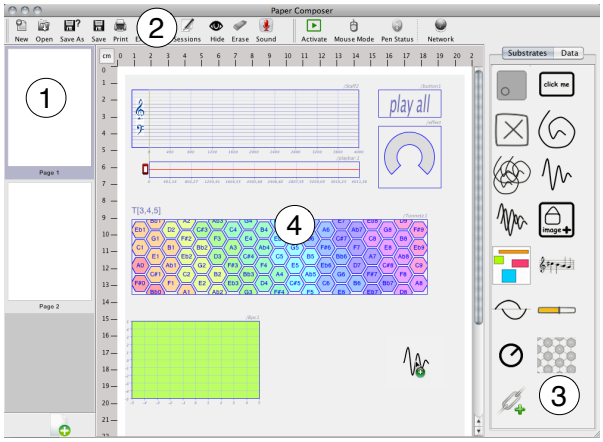

Figure 1: PaperComposer interface
(1) Current document pages.
(2) Toolbar to manage document and set parameters.
(3) Thumbnails of available component classes can be dragged into the virtual page to create new instances.
(4) Virtual page with components: musical staves, playbar, button, knob slider, tonnetz, curve container.

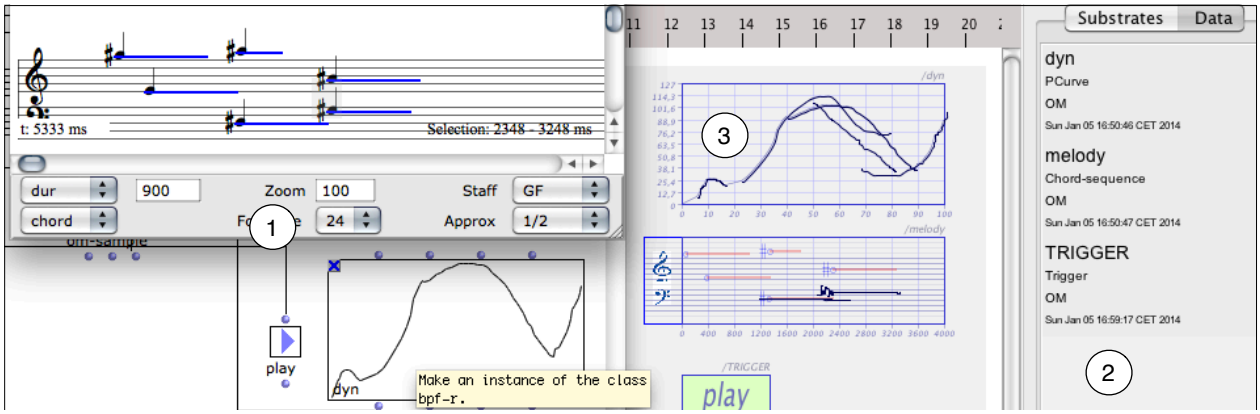

Figure 2: Synchronizing data between an external music tool (*OpenMusic*) and *PaperComposer*.
(1) OpenMusic patch with a graph and chord sequence, exportable via a custom *OpenMusic* library.
(2) List of imported data from OpenMusic that can be dragged to any component.
(3) Graphical representations of the components, data and strokes drawn on paper.

To achieve these goals, we developed *PaperComposer*, an interface builder for creating paper components that represent *paper substrates* and control digital musical classes. Users can create specialized components with various properties and connect them via OSC data channels to external applications. They can combine multiple components into custom page layouts and test the interaction with a mouse before printing the page on paper with the Anoto dot pattern. *PaperComposer* provides live feedback as the user uses an Anoto pen to draw on the page, and enables the user to store and reuse previously drawn pen strokes.

Document Creation and Customization

Figure 1 illustrates the creation of a two-page document in portrait format. The user can add components by dragging them from a list at the right side of the page window. The user can resize and position a component on any page and then parameterize it through a configuration window activated with a right click. Parameters that can be reconfigured include the color of the component and its output OSC channels as well as component-specific parameters, such as a range of values and duration, if the component represents a musical sequence. The user can define static links between components represented by straight non-printable lines. Components can accept data from compatible data sources, both files and external applications.

Figure 2 shows how data exported from OpenMusic appears in a list in *PaperComposer*. In this scenario, the user decides to edit an existing musical sequence in OpenMusic by adding new notes, modifying the duration of existing notes and controlling their dynamics with an amplitude curve. The user exports three different data streams from OpenMusic: a curve, i.e., a list of points, a musical sequence, and a trigger to consume events such as a pen down to launch the play command. The user can establish an association by dragging any data source element from the list and then dropping it onto a compatible component. If the user drops the data onto a blank part of the page, a menu appears with all compatible components. The user makes a choice, which creates a new paper substrate that hosts the associated data.

The *PaperComposer* components that appear on the screen can be printed directly onto interactive paper with an automatically superimposed Anoto dot pattern. *PaperComposer* also supports the creation of interactive paper interfaces from existing scores created with musical editors such as *Finale* and *Sibelius*. Each page of the score becomes a new document page with associated interactive regions for each score element. For example, each note is interactive. When the pen touches a note, PaperComposer plays the corresponding pitch.

Deployment, Debugging and Session Management

PaperComposer displays the users interactions on paper in real time on the screen, including the current page and the strokes being drawn on paper. Figure 2 shows the paper interface at runtime, overlaid with the current strokes as well as how the associated musical objects were updated in OpenMusic.

All strokes and audio recordings captured during a session are automatically stored and labeled for later reuse. The user can use the session manager from the menu bar to manage past sessions, load them onto the screen and use components to reinterpret them. Developers can also test newly created components with data of older sessions and iteratively improve their components by reusing recorded strokes as input. Finally, *PaperComposer* supports the monitoring of the network activity between the paper application and musical software. The user can open a monitor window by clicking on the network icon of the top toolbar and track connected client applications and OSC messages exchanged through the network.

Examples of Interactive Components

Figure 3 gives an overview of paper components currently available in *PaperComposer*.

1. *Knob slider*: controls continuous values in real time.

2. *Button*: triggers actions in response to one or more pen events: down, up, moved, entered, exited, tap, and double tap.
3. *Curve container*: supports entering curves and provides a customizable grid layout to reinforce different levels of precision. One type of curve container supports entering a single curve such that each new stroke is merged to the curve, incrementally changing its trace. The other type supports creating collections of curves, such that each new stroke represents a different curve.
4. *Musical sequence container:* supports interaction and editing of timeline representations of musical sequences [8].
5. *Playbar*: selects time ranges to play specific parts of a musical sequence.
6. *Tone network:* supports the interactive creation of musical sequences of both chords and melodies [3].

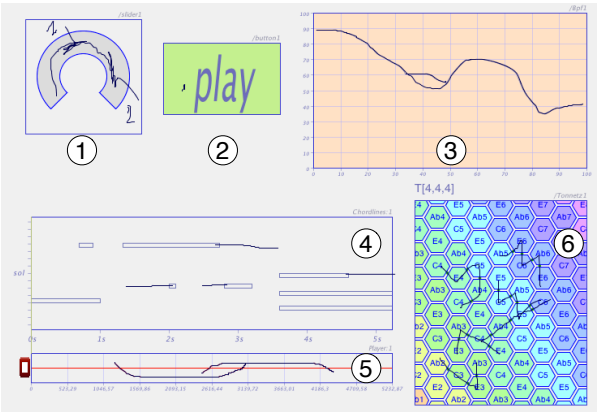

Figure 3: Examples of existing components.

AN API FOR PROGRAMMING PAPER COMPONENTS

We created an application-programming interface (API) that provides classes for extending *PaperComposer* with new data representations and interactive components. It includes classes for creating and printing customized templates, processing typed data, storing data and sessions, handling pen events and pen gestures, and receiving or sending messages to external applications. Figure 4 summarizes the architecture of the API.

Main API Components

The API allows the developer to easily create paper applications by defining instances of existing classes of paper components on paper coordinates of one or more pages. The developer can also create new types of components by extending the *PComponent* class. The API takes care of the management of the Anoto dot pattern, the management of sessions, and the communication of the components with external applications. We use *Java Reflection* to load new component definitions and their instances at run time. Multiple components can be grouped together into a single paper application represented by the Document class. A document specifies a set of pages, where each page can contain several components. We have developed utilities to store, edit and load a single or multiple documents for live interaction. All pen gestures created by users at runtime are stored within sessions for later reuse and archiving purposes. In addition, the API supports recording audio synchronized with the captured pen data.

A component is a composite paper widget, similar to the first-class widgets supported by common user interface toolkits such as Java Swing. It has a shape and position on paper expressed in millimeters and has a visual representation on screen and on paper. More specifically, it provides methods responsible for the painting and printing of its graphical elements and data. These methods are compliant with *Java2D*, letting developers reuse existing *Java2D* code. A component can be associated with coordinate transforms that map its data from millimeters to application-specific values. It can also register to listeners of specialized pen events such as pen down, pen entered, stroke created, or gesture recognized that happen within its shape.

A component can contain one or more interactive components, where each component can have its own listeners. Printed and handwritten data can also be added as child components that listen to pen events. Since printed pages may contain large numbers of components that respond to pen events, we developed an event manager to store and retrieve components and data by using efficient spatio-temporal indexing techniques. We use the R-Trees implementation by the SaIL Library [10].

Communication among Paper Substrates

We designed communication mechanisms that let us create complex musical data from several layered or connected components. Components can communicate with each other through static and dynamic links [8]. Static links are created either programmatically or by using the graphical interface of PaperComposer. Dynamic links are created interactively with the pen by drawing a stroke that crosses two different components.

Users can only create links among compatible components. One of the linked components usually acts as a master that defines the reference of data sharing between them. The link mechanism also enables a paper component to reuse strokes written on other components, reinterpreting, rescaling and transforming them before redirecting them to specialized patches on the computer.

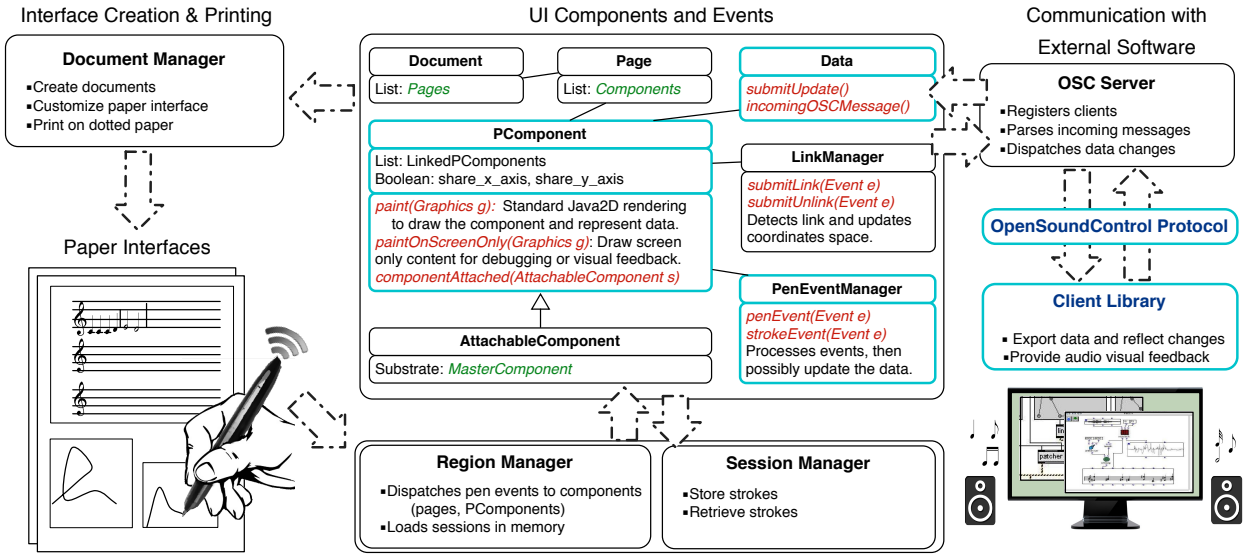

Figure 4: *PaperComposer* programming API supports creation of paper interfaces to manipulate musical data. Developers edit blue elements to identify data types, specify data representations and define user interactions.

Communication with External Music-Programming Software

Components can exchange musical data with external applications using the OSC protocol [23]. An OSC server allows registered applications to send and receive updates from components. We define an OSC-based data exchange protocol for a variety of data types, from basic lists of numerical values to more complex sequences music notes and chords. Data types are extensible and developers can add new data types or implement new components to handle them. *PaperComposer* can connect components with *Max* and *OpenMusic* libraries, chosen because they widely used by composers and can be easily extended programmatically. The *OpenMusic* library is implemented in *Common Lisp* and supports common musical objects such as graphs for controlling musical parameters and basic score containers. These objects can export their data to interested components. The *Max* library is written in *JavaScript* as an external object and can be used within any *Max* patch to export its window and interface elements, e.g., buttons, graphs and sliders. Developers and users can extend the two libraries to add new data types and functionality.

Support for Musical Scores

The *PaperComposer* API also supports the use of scores from commonly used music notation editors such as *Finale* and *Sibelius*. A dedicated class loads the formal description in the *MusicXML*[2] format with the PDF of the score. Each graphical element of the score is associated with an interactive area on the paper page and related to the corresponding musical content, using the *ProxyMusic* Library[3]. We also created a class to play the corresponding MIDI note if the pen enters a particular note's area. Components can also use the data from the score to create alternative representations and export the result into a new *MusicXML* file that can be processed and rendered by most score editors.

Component Creation Process

Developers follow these steps to implement and deploy a new paper substrate or build their own interface:

Step 1: Create a class that inherits the *PComponent* class or the *AttachableComponent* class.

- Set an icon file and a name.
- Set the coordinate space and set Booleans to indicate how axes should be synchronized when links with other components occur. This allows a component to use the axis and scale of a linked master component and apply it to its data.
- Set the accepted data types.
- Program the paint method with standard *Java2D* instructions.
- Add one or several pen event listeners and implement how the events modify the data.

Step 2: Create a class that inherits the *Data* class.

- Define an OSC communication protocol.
- Implement *incomingOSCMessage()* & *submitUpdate()* methods.

Compatible data sent by an external application can be loaded into any component. Touching a component with the pen triggers a data notification to the application.

We provide several utilities to simplify stroke recognition, including using a distance threshold, Douglas and Peucker's algorithms [5], and extracting geometrical features, including main orientation, duration, intersections, from other strokes. Interpreting pen input remains challenging due to the high variability in how users write

[2] http://www.musicxml.com/

[3] https://proxymusic.kenai.com/

with respect to temporal order and shape. *PaperComposer* supports the use of *Java* libraries and available toolkits such as *iGesture* [19] to recognize gestures.

EXAMPLES OF PAPERCOMPOSER APPLICATIONS

We used the *PaperComposer* API to design and implement several interactive paper applications.

Tangible Scores for Electro-Acoustic Composition

We explored how physically positioning interactive pieces of paper can be used to create musical sequences. We developed a tangible score application that lets users create musical sequences by assembling several sounds on a paper interface.

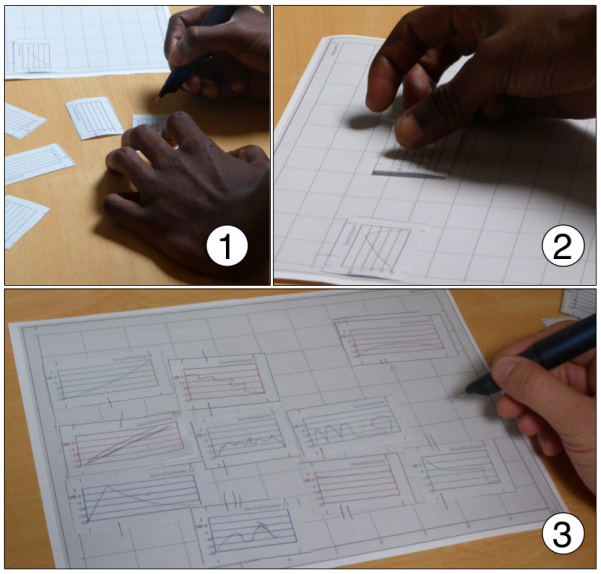

Figure 5: Creating a score with movable substrates.
(1) Draw a curve on a sound substrate to control amplitude.
(2) Position a sound substrate to define onset time.
(3) Touch the score substrate to listen to the resulting score.

Interactive paper design: The application includes a paper substrate that takes a whole page and represents the space of the score where other small substrates linked to sound files can be added. Each small substrate has a color that identifies a sound and a width that defines its duration. The user can draw curves control the amplitude of the sound over time. The user can also assemble the small elements into the main sequence by sticking them onto its 2D space and creating a dynamic link between the two substrates with the pen. This creates a collage of sounds. The user can tap the pen on any individual paper substrate to play the associated sound or tap on the score to hear the complete piece. The paper interface communicates with the *Max* patch responsible for amplifying and mixing the sounds based on instructions given by the paper substrates. This application is a simplified but tangible representation of *OpenMusic*' [4] maquette interface, which provides a spatial layout to help musicians mix programmable musical objects together and organize them over a timeline.

Public demonstration: We demonstrated the user interface during the *Fête de la Science 2011* (see Figure 5). We encouraged participants to explore the properties of different sounds and understand the goals and tools of computer-assisted music analysis and composition.

Approximately 60 to 70 participants, mostly children, interacted with the interface to create collaborative musical pieces. They enjoyed editing their scores with the pen. Several explored alternative results by spatially rearranging their sound components and by drawing different curves on *paper substrates* of the same color. We observed that the tangible nature of the interface is particularly useful for making music creation tools accessible to children in a playful manner.

Proportional Musical Notation

We collaborated with a graduate student in composition at Ircam who had created a personal musical notation for a composition for alto and electronics. Using a participatory design approach, we used *PaperComposer* to iteratively design a *paper substrate* that recognizes handwritten notes over a staff, lets him to listen to the musical result and transfer it to *OpenMusic* or *Max*.

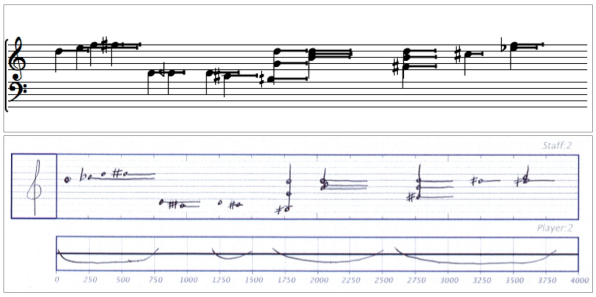

Figure 6: *Paper substrate* co-designed with a composer.
Top: Resulting musical score in a *Max* patch.
Bottom: Handwritten notes represent pitch and duration.
Playbar substrate selects, evaluates individual score segments.

Figure 6 presents the score written by with his personal notation and the resulting musical sequence in *Max*. According to this notation, notes have an absolute position in time, which is defined by their horizontal position within the staff. A horizontal line nested to a note represents its duration. Our recognizer is based on simple heuristics and supports the following elements: keys (G, F and C alto), notes, intonation symbols (quarter tones, sharps and flats) and duration lines. The composer's original notation does not include stems, i.e., vertical bars that group notes into chords. We added stems to simplify the identification of chords and their onsets. The user can evaluate the musical result by tapping on the rectangular area around the key symbol or by using an additional playbar substrate. To correct a recognition error, the user can hold the pen over the misrecognized symbol and redraw it.

The composer created his own interface components with *PaperComposer*. He combined the notation substrate with several graph substrates to control the dynam-

ics of his musical sequences. He successfully created *Max* patches that react to the data transmitted from the pen to produce a final sequence played by his own synthesizer.

Recognizing handwritten musical notes is a challenging problem [15]. Previous work has tried to tackle it by using simplified gesture vocabularies [6] or interactive devices that allow the user to supervise the recognition process and correct errors [21]. The present *paper substrates* considerably simplify the recognition problem by using a single type of note with no special notation for rhythms or rests.

Controlling Sound Synthesis with Pen Gestures

We collaborated with another composer to design a *paper substrate* for drawing gestures that control a sound synthesis engine developed by the composer in *Max*. According to his composition framework, the visual space acts as a sequencer where the horizontal axis represents time. The composer organizes several musical elements in time by positioning them in the patch space. He mainly uses text-based descriptions of synthesis parameters but is interested in enriching his expression possibilities with pen input.

We worked with the composer to create a *paper substrate* that maps dynamic gestures drawn on paper to the dimensions of computerized objects in a *Max* patch. The actual properties that form the profile of a gesture are the list of its point coordinates with their timestamps and the associated pen pressure values. We use color-coding to reveal the orientation of a gesture, and visual information that the composer uses in the synthesis engine. The beginning of each stroke appears as blue and progressively changes to red.

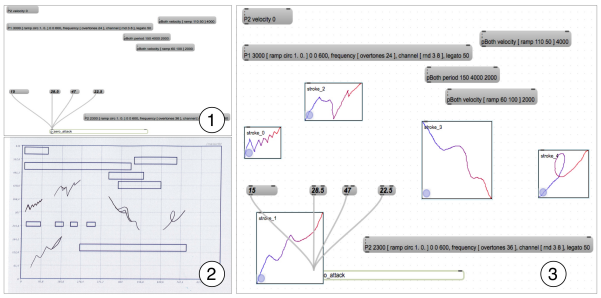

Figure 7: Controlling sound synthesis with pen gestures.
(1) Original score in a *Max* patch.
(2) Score substrate with handwritten gesture refinements.
(3) Resulting score in a *Max* patch.

The *paper substrate* represents existing score objects as rectangles that guide the drawing of the synthesis gestures. After expressing a gesture on paper, the composer can refine it by redrawing only parts of the gesture. In this case, the refining stroke is merged with the initial one. We carefully update the timestamps of the new data points so that they preserve the dynamics of the original gesture. This allows the composer to explore variations of a gesture's shape without changing its original dynamics. He can also edit the position of a gesture in the electronic view and then reprint an updated version of the *paper substrate* in order to continue his work on paper.

Figure 7 illustrates how the composer uses the *paper substrate* as part of his workflow. First, he exports sequences of musical objects from his *Max* patches. He then uses *PaperComposer* to associate the exported data streams with a specialized sequence substrate. He prints the interface on paper to draw his synthesis gestures. The *Max* patch shows the new gestures immediately after they are drawn on paper.

DISCUSSION AND FUTURE DIRECTIONS

PaperComposer supports the implementation and deployment of powerful user interfaces for working with music on paper. We observed that linking music programming software with paper applications could make such environments more accessible to children and novice users. At the same time, our approach can help professional composers reflect on the implementation of their ideas on paper while interacting with their composition tools.

PaperComposer requires the user to define the whole interface on the screen before printing and using it. However, studies have shown that composers frequently use paper interfaces for ill-formed ideas and iteratively refine both their ideas and their notation [7, 12, 16]. Therefore, we also need to consider the interactive definition of *paper substrates* by drawing them directly on paper before or after writing musical content. For example, the Livescribe pen allows users to draw a piano keyboard made of vertical and horizontal lines anywhere on the page. Once the piano is drawn, the pen can play the corresponding note when the pen tip enters a key. We are interested in extending our tools to offer similar functionality by enabling users to define interactive structures, such as an interactive musical staff, by drawing directly on paper. This may help composers better integrate paper interfaces in their composition process with incremental transitions between ill-formed ideas and more formal ones.

The PaperComposer toolkit provides assistance for programming new components and for creating and customizing end-user interfaces. Composers can build personal interactive-paper interfaces for a range of music-creation tasks: drawing and refining sound-synthesis curves, writing musical sequences by using music notations or Tonnetz representations, controlling music software with paper buttons, timeline selectors, playbars and sliders.

To extend the collection of paper substrates supported by PaperComposer, we met with composers and examined the representations that they use to write music. We collaborated with them to design new *paper substrates* that assist or enhance specific aspects of their work process. We implemented these *paper substrates* with our toolkit. Composers could then use them within *PaperComposer* to connect their paper-based activities with their comput-

er-aided composition environments. More recently, we used the toolkit to create *Polyphony*, which was tested on a short composition task by 12 music composers [9].

Currently, we support only audio and visual feedback via external applications in response to events and data transmitted by the pen. However, we are also interested in supporting additional feedback modalities and richer forms of physical interaction with pen and paper, such as bimanual interaction techniques based on the use of portable mini-projectors [20] and mobile phones [21]. Such approaches can bring visual feedback closer to the space of writing. We are also interested in extending the API and *PaperComposer* to enable programmers and users to define or reconfigure such forms of audio and visual feedback at the level of the actual paper components. This will result in richer paper applications that mix both the physical and the virtual instances of a paper component. A longer-term goal is to explore the dynamic definition of new interactive substrates, either on paper or on the screen, with visual programming languages, since many composers are familiar with such languages.

SUMMARY AND CONCLUSION

We developed a toolkit that includes a user interface builder, *PaperComposer*, and an API. *PaperComposer* allows both developers and end-users to create interactive paper interfaces with specialized paper components, connect the components with data streams of external musical software, customize their appearance and deploy their interactive paper applications. It also provides mechanisms for session management, debugging, and logging. The API enables developers to define new paper components that accept additional musical data with their own representation structures and interactions.

We used these tools to create new paper interfaces for musical creation. We developed and demonstrated a paper application for creating electro-acoustic scores based on the spatial positioning of movable paper components. We also collaborated with composers to design interfaces for writing music notation and controlling sound synthesis. These examples demonstrate the practical use of our toolkit. Future work needs to further evaluate it through workshops with developers of music applications. We are also interested in using it in longer field studies with professional composers and musical assistants.

ACKNOWLEDGMENTS

We are grateful to all the composers who helped us improve *PaperComposer*. We would also like to thank the researchers from the Musical Representations team at IRCAM and the In|Situ| team at LRI for their help and insights. This work was supported in part by ANR project INEDIT (ANR-12-CORD-0009).

REFERENCES

[1] Agon, C., Bresson, J. and Assayag, G. 2006. *The OM composers' book Vol.1 & Vol.2*. Collection Musique/Sciences.

[2] Arai, T., Aust, D. and Hudson, S.E. 1997. PaperLink: a technique for hyperlinking from real paper to electronic content. In *Proc. ACM/CHI '97*, 327–334.

[3] Bigo, L., Garcia, J., Spicher, A. and Mackay, W.E. 2012. PaperTonnetz: Music Composition with Interactive Paper Tonnetz. In *Proc. SMC'12*, 219–225.

[4] Bresson, J. and Agon, C. 2008. Scores, Programs, and Time Representation : The Sheet Object in OpenMusic. *Computer Music Journal*. 32-4, 31–47.

[5] Douglas, D.H. and Peucker, T.K. 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*. 10, 2, 112–122.

[6] Forsberg, A., Dieterich, M. and Zeleznik, R. 1998. The music notepad. In *Proc. UIST '98*, 203–210.

[7] Garcia, J., Tsandilas, T., Agon, C. and Mackay, W. 2011. InkSplorer: Exploring Musical Ideas on Paper and Computer. In *Proc. NIME '11*, 361–366.

[8] Garcia, J., Tsandilas, T., Agon, C. and Mackay, W. 2012. Interactive paper substrates to support musical creation. In *Proc. ACM/CHI '12*, 1825–1828.

[9] Garcia, J., Tsandilas, T., Agon, C. and Mackay, W.E. 2014. Structured observation with polyphony: a multifaceted tool for studying music composition. In *Proc. ACM/DIS '14*, 199–208.

[10] Hadjieleftheriou, M., Hoel, E. and Tsotras, V.J. 2005. SaIL: A Spatial Index Library for Efficient Application Integration. *GeoInformatica*. 9, 4, 367–389.

[11] Hall, P. and Sallis, F. 2004. *A handbook to twentieth-century musical sketches*. Cambridge University Press.

[12] Healey, P.G.T., Jean-Baptiste Thiebaut and Thiebaut, J.-B. 2007. Sketching Musical Compositions. In *Proc. CogSci' 07*, 1079–1084.

[13] Heinrichs, F., Steimle, J., Schreiber, D. and Mühlhäuser, M. 2010. Letras: an architecture and framework for ubiquitous pen-and-paper interaction. In *Proc. ACM/EICS '10*, 193–198.

[14] Hurter, C., Lesbordes, R., Letondal, C., Vinot, J.-L. and Conversy, S. 2012. Strip'TIC: Exploring Automatic Paper Strip for Air Traffic Controllers. In *Proc of AVI '12*, 225.

[15] Lee, K.C., Phon-Amnuaisuk, S. and Ting, C.Y. 2010. A comparison of HMM, Naïve Bayesian, and Markov model in exploiting knowledge content in digital ink: A case study on handwritten music notation recognition. *IEEE International Conference on Multimedia and Expo*, 292–297.

[16] Letondal, C., Mackay, W.E. and Donin, N. 2007. Paperoles et musique. In *Proc. ACM/IHM*, 167–174.

[17] Newman, T.U. 2008. *The Creative Process of Music Composition: A Qualitative Self-study*. ProQuest.

[18] Norrie, M.C., Signer, B. and Weibel, N. 2006. General framework for the rapid development of interactive paper applications. In *Proc. CoPADD*. 6, 9–12.

[19] Signer, B., Kurmann, U. and Norrie, M. 2007. iGesture: a general gesture recognition framework. In *Proc. ICDAR '07*, 954-958.

[20] Song, H., Guimbretiere, F., Grossman, T. and Fitzmaurice, G. 2010. MouseLight: bimanual interactions on digital paper using a pen and a spatially-aware mobile projector. In *Proc. ACM/CHI '10*, 2451

[21] Tsandilas, T. 2012. Interpreting strokes on paper with a mobile assistant. In *Proc. UIST '12*, 299.

[22] Tsandilas, T., Letondal, C. and Mackay, W.E. 2009. Musink: composing music through augmented drawing. In *Proc. ACM/CHI 09*, 819–828.

[23] Wright, M. and Freed, A. 1997. Open Sound Control: A New Protocol for Communicating with Sound Synthesizers. In *Proc. ICMC'97*, 101–104.

[24] Yeh, R.B., Paepcke, A. and Klemmer, S.R. 2008. Iterative design and evaluation of an event architecture for pen-and-paper interfaces. In *Proc. UIST '08*, 111.