Software for Interactive and Collaborative Creation in the Classroom and Beyond: An Overview of the Soundcool Software

Stefano Scarani,* Adolfo Muñoz,† Jaime Serquera,§ Jorge Sastre, and Roger B. Dannenberg††**

*Department of Sculpture
†Institute of Design and Manufacturing (IDF)
**Institute of Telecommunications and Multimedia Applications
*†**Universitat Politècnica de València
Camino de Vera s/n, 46022 Valencia, Spain
§Conservatorio Superior de Música "Salvador Seguí" de Castellón
Carrer del Marqués de la Ensenada 34–36, 12003 Castelló de la Plana, Spain
††School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, Pennsylvania 15213, USA
sscarani@musikene.net, amunyoz@upvnet.upv.es, jaiserpe@upv.es, jsastrem@upv.es, rbd@andrew.cmu.edu

Abstract: This article presents a free framework for collaborative creation of interactive and experimental computer music called Soundcool. It is designed to fill a gap between rigid ready-to-use applications and flexible programming languages. The system offers easy-to-use elements for generating and processing sound, much like ready-made applications, but it enables flexible configuration and control, more like programming languages. The system runs on personal computers with an option for control via smartphones, tablets, and other devices using the Open Sound Control (OSC) protocol. Originally developed to support a new music curriculum, Soundcool is being used at different educational institutions in Spain, Portugal, Italy, and Romania through EU-funded Erasmus+ projects. In this article, we present our system and showcase three different scenarios as examples of how our system meets its objectives as an easy-to-use, versatile, and creative tool.

Introduction

Soundcool (http://soundcool.org) is a computer music software system for live performance. It emphasizes the use of mobile devices as controllers, collaboration through the use of multiple control devices, and configurable high-level software components for computer music generation and sound processing. Soundcool was originally designed to support a music curriculum for primary and secondary school students who collaboratively create experimental computer music (Sastre et al. 2013, Sastre et al. 2015). The Soundcool design has emphasized ease of use from the beginning. Soundcool has found application in classroom activities, in student concerts in auditoriums and other public venues, and for live electronics in a new opera (http://themotheroffishes.com). Whereas previous publications have discussed Soundcool for education, this article discusses some of the design features of Soundcool that we believe are of general value, and we also discuss Soundcool as a tool for advanced compositions and performances.

We believe there is a gap between off-the-shelf, ready-to-use applications and programming-oriented software development platforms. We will discuss these approaches and then present Soundcool as a compromise that achieves a balance between these poles. Key features of Soundcool are: self-contained, patchable modules for audio generation and manipulation; control extensibility supporting a variety of interface devices and languages; and

Computer Music Journal, 43:4, pp. 12–24, Winter 2019
doi:10.1162/COMJ_a_00534

extensibility through audio plug-ins. We show the importance of these features in several Soundcool applications.

Ready-to-Use Music Applications

Examples of ready-made applications are audio effects processors, sample-playback systems, and sequencers (Manzo 2015). Additionally, a great variety of music apps have been developed in the past few years for tactile mobile devices (see, for instance, Stuart Dredge's overview of music-making apps for beginners in the *Guardian*, 17 October 2005). These applications are popular because they provide fully functional tools for musicians. All of these applications offer the possibility of modifying some of their configurations, but they are, by definition, designed to serve a particular role.

Ready-made applications are convenient when they offer the right solution, but often the interfaces and options in music software are highly normative, with narrow assumptions about how music should be structured and performed. This defeats one of the main attractions of computing, which is the idea of the "universal machine" that can, through software, accomplish any task and offer any behavior. To branch "outside of the box," or simply to achieve extraordinary generality, one must resort to programming to develop more customized solutions.

Programming-Oriented Music Software Platforms

Software development platforms such as Max, Pure Data (Pd), Csound, Nyquist, Supercollider, and ChucK offer users free rein to build virtually any musical devices they can imagine (Dannenberg 2018). Although these systems differ along some important dimensions (real-time interactivity, textual versus visual programming, data structures, abstraction), they have in common the ability to specify computations by combining lower-level operations. In general, programming-oriented systems are more technical than ready-made applications, so considerable expertise and experience are necessary to make the most of these systems.

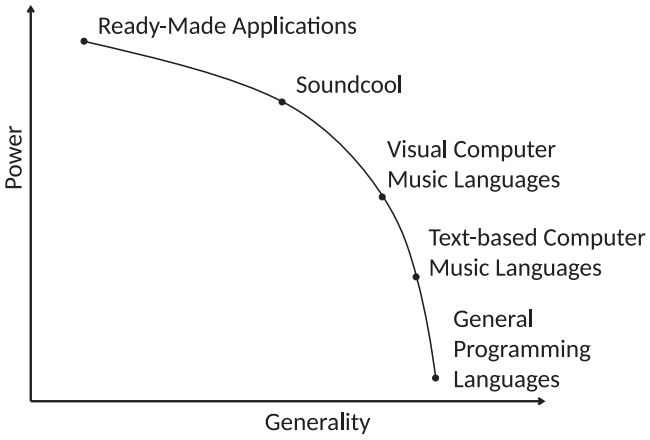


Figure 1. Power (i.e., ease of use) versus generality (i.e., the ability to accomplish a wide range of creative tasks). In these terms, the ideal system is both powerful and general (upper right), but there seems to be a trade-off.

Power versus Generality

Borrowing some terminology from early AI research (Newell 1983), we can say that systems are more general when they solve a greater variety of problems, and more powerful when they are particularly adept at solving a particular problem. Ready-made applications are powerful, because they require little effort to accomplish tasks; but because they focus on a specific approach, applications lack generality to address a wide range of tasks or conditions. At the risk of over-simplifying, Figure 1 offers an intuitive perspective on the spectrum of computer music systems from ready-made applications to general-purpose programming languages. For any given task, a user has the choice of using a ready-made solution (if one exists) or creating a solution using increasingly more general but more effortful programming languages.

For computer music tasks, specialized computer music languages are often sufficient, avoiding the need to build everything from scratch. If users benefit from specialized languages, how much further can we reduce generality in pursuit of still greater power? What are the interesting points of balance between these two objectives? Soundcool explores the region approaching maximal power while remaining quite programmable and configurable. It is not the only

example of this design approach, but it is interesting because of its availability and years of evolution, mainly in the context of music education.

Related Work

Many works point to the pedagogical potential of new technologies and new human interfaces (Lloret Romero 2007; Sánchez, Salinas, and Sáenz 2007; Savage 2007; Clough et al. 2008). A number of technologies are presently emerging from STEAM initiatives within primary and secondary level music education. (STEAM extends STEM—science, technology, engineering, and mathematics—by incorporating arts into the equation.) It is believed that arts education is fundamental to foster creativity, and in turn, creativity is key to innovation in every industrial area. Another trend in educational initiatives is collaborative environments (Baumann et al. 2011). Regarding music technologies within STEAM education, EarSketch is a noteworthy development. EarSketch is software along with a curriculum that motivates learning and applying computer science concepts through music remixing. As their authors describe,

> . . . students learn to code in JavaScript or Python, tackling learning objectives in the Computer Science Principles curricular framework as they simultaneously learn core concepts in music technology. They create music through code by uploading their own audio content or remixing loops in popular genres created by music industry veterans (Freeman et al. 2015, p. 1).

Thus, EarSketch is more oriented toward music production and algorithmic control, while Soundcool is oriented more toward collaborative music performance.

Another example of a STEAM-oriented learning system is BlockyTalky (Shapiro et al. 2017, p. 53). This system describes a toolkit for "distributed and physical computer music systems-building and performance." BlockyTalky encourages young students to create communicating systems using a block-based language to control software synthesizers. The authors comment on the hazard that students gravitate toward writing sequential programs to reproduce linear, note-based, noninteractive work, missing out on creative opportunities offered by technology. In general, BlockyTalky instruction seems to emphasize music to motivate programming. BlockyTalky has more focus on designing control strategies whereas Soundcool focuses on sound design and performance.

Soundcool might be compared to AudioMulch (www.audiomulch.com), which also features high-level processing modules and a graphical, patch-oriented interface. Both systems emphasize audio configuration by patching while control is accomplished through built-in graphical interfaces for each module. Flexibility is provided through external control via MIDI (AudioMulch) or Open Sound Control (Soundcool). Soundcool is generally simpler, however. For example, Soundcool modules have predefined interfaces to mobile devices that are easily enabled and immediately usable, whereas control parameters in AudioMulch must be set up individually by hand. AudioMulch has a separate detailed control panel for each module, whereas Soundcool's modules are represented directly by their interfaces.

Cycling 74's BEAP (cycling74.com/tutorials/a-few-minutes-with-beap-tutorial-series) is a collection of preconstructed Max patches complete with interfaces, which makes BEAP at least comparable to Soundcool. BEAP is modeled after modular analog synthesizers, so users must understand the idea of control voltages and must route control as well as audio signals. Users must also master at least the basics of the Max interface and cope with Patching and Presentation modes and other details, and BEAP is not designed for collaborative projects. Hans Tutschku's 264 Tools (github.com/mus264/264-tools) is another collection of synthesis modules for Max similar to BEAP, and Automatonism is a comparable system for Pd (www.automatonism.com).

One could also look to products like Reaktor (www.native-Linstrument.com/en/products/Komplete/synths/reaktor-6), Guitar Rig (https://www.native-instruments.com/en/products/komplete/guitar/guitar-rig-5-pro), and Mainstage (www.apple.com/mainstage) for patchable modules

with preconstructed graphical interfaces. In fact, effects chains in digital audio workstations are another example of patchable modules, and they offer a great deal of flexibility. One drawback, at least from an educator's perspective, is that commercial products tend to be loaded with features to attract customers and address a variety of requirements; they tend therefore to be too complex for youngsters and beginners.

In addition, few existing products or research systems are explicitly designed to support collaborative performance. One interesting software system is MobMuPlat (www.mobmuplat.com), with which one can combine graphical interface design and Pd patches for execution on iOS or Android systems. This system offers ample interesting functionality, especially for electronic ensembles, but it is intended for experienced Pd users.

Finally, we should mention libraries as another approach to move up the power-versus-generality curve. Libraries extend a programming language with reusable modules, making many tasks easier to accomplish. For example, BEAP is essentially a library within Max, giving users powerful, high-level modules while retaining more general capabilities for when they are needed. Libraries certainly add power to the underlying language, but, ultimately, the use of libraries requires users to write lower-level "glue" code and use development environments that are more complex, and this can be especially difficult for young students and casual users.

The Soundcool Approach

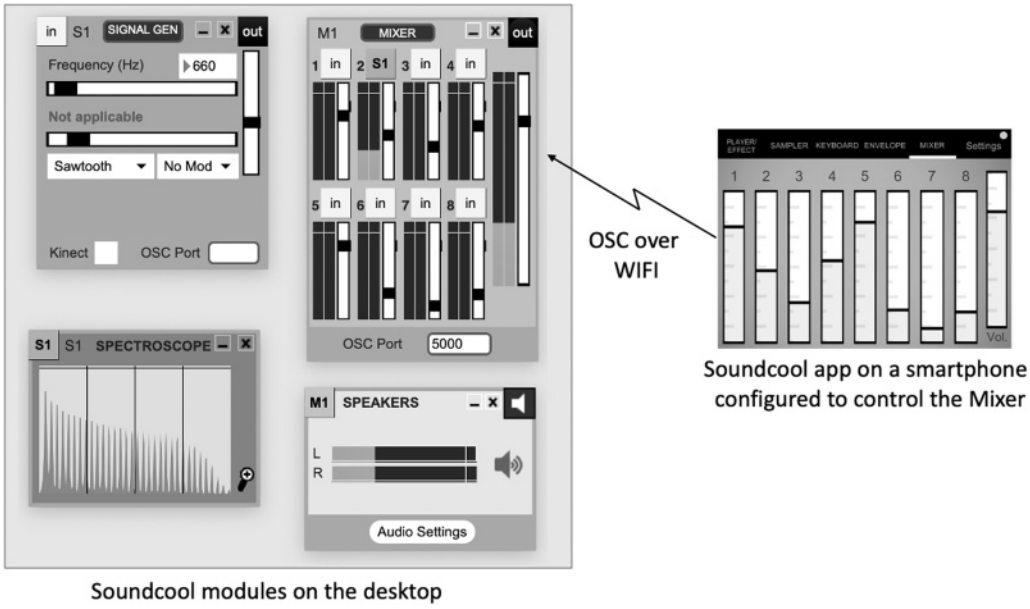
With all these possibilities and decades of computer music system development, why would we consider yet another system? Soundcool was motivated by the desire to create a platform for a new approach to music education for children. In particular, Soundcool prioritizes the possibilities of digital media as opposed to traditional instruction and emphasizes creativity and collaboration as opposed to theory and instrumental performance. We will describe Soundcool's role in education in greater detail herein.

Soundcool offers a set of modules that run on a central host computer. Figure 2 illustrates a simple Soundcool program or patch running on a laptop and a controller running on a smartphone. Each module can be considered as a musical instrument, such as a synthesizer, a sampler, a sound effect, etc. Soundcool modules can be interconnected in different ways allowing users to create their own computer music systems. Additionally, modules can be controlled with the mouse on the host computer or remotely using mobile devices and Open Sound Control (OSC). Typically, students operate mobile devices in collaborative performances, with each student in charge of at most one of the sound modules on the host computer. The OSC component makes Soundcool an open system capable of receiving control messages from other pieces of programming software, as we describe in the next sections.

Although created for primary and secondary school teachers and their students, Soundcool has proven useful for a variety of other applications. We believe that Soundcool's success is due to a combination of factors, and a better understanding of these could benefit the design of other computer music systems. One contributing factor to generality is support for audio plugins, allowing users to choose among hundreds of sophisticated audio processors (compare this to a fixed set of audio unit generators, even in a computer music programming language). Another factor is access to control parameters over OSC, which allows users to extend Soundcool using almost any programming language and operating system. A third factor is a visual programming paradigm in which modules are represented by ready-made graphical user interfaces, minimizing the effort to create, understand and manipulate sound processing systems.

Soundcool is not unique to offering high-level modules, audio plugins, or OSC interfaces. However, we claim that it is the combination of these factors that leads to an interesting tradeoff in power and generality useful to a wide spectrum of users, from music teachers to professionals. We support this claim by describing several applications of Soundcool.

Figure 2. A simple Soundcool patch running on a laptop with a touch-screen interface running on a smartphone.



In the next section, we describe Soundcool in more detail. Then we describe three Soundcool applications: music education, algorithmic control of sound synthesis, and telematic performance. We follow with insights we have gathered and make suggestions for the design of future computer music systems.

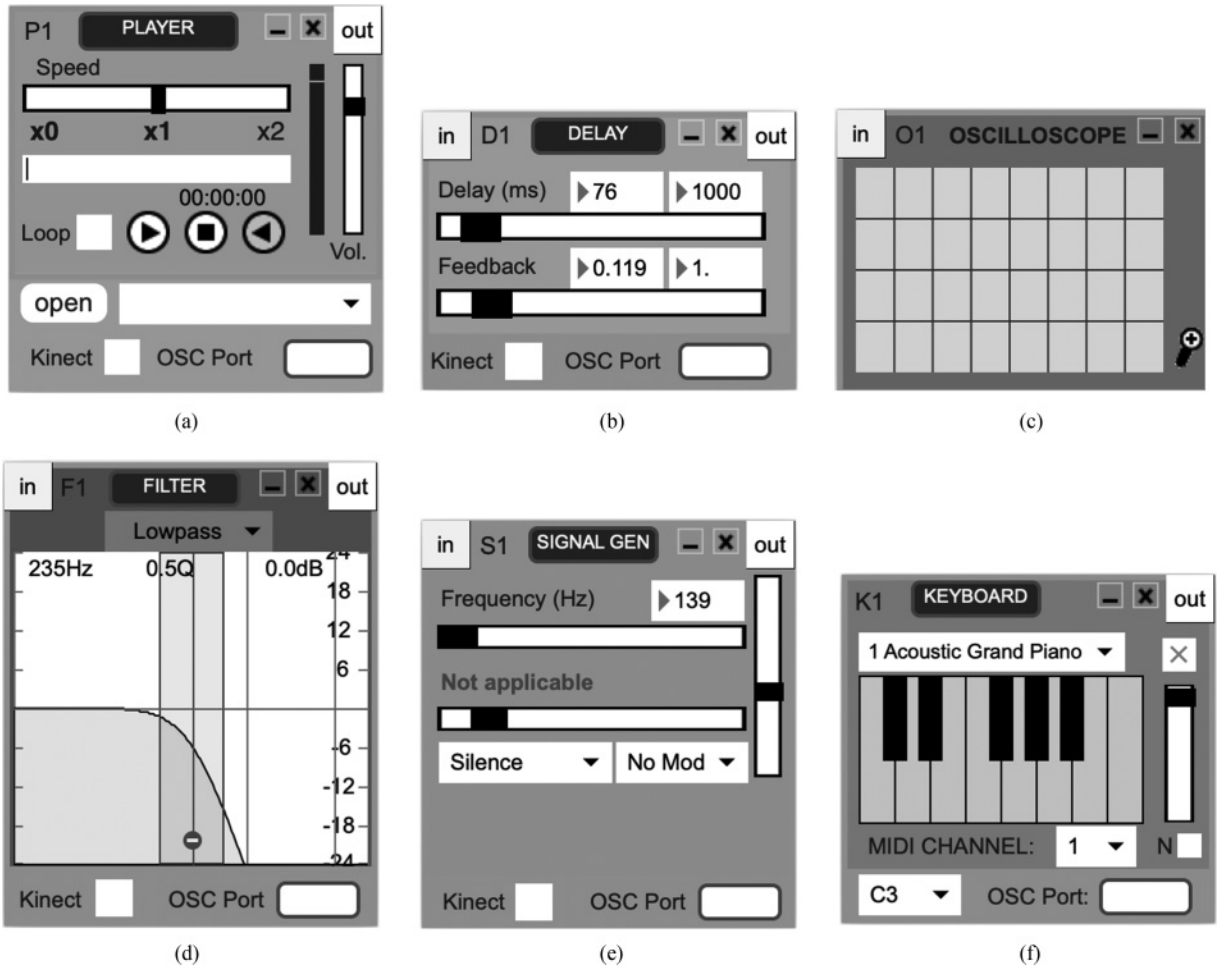
Soundcool Overview

One important design constraint was to make Soundcool inexpensive to incorporate into classrooms. Besides being free and open-source (available at soundcool.org/en/downloads), it works on any Mac OS or Windows computer with its own speakers or headphones and microphone. Typically, classrooms have at least a single computer, and students have their own smartphones or low-cost tablets, which serve as distributed multitouch control surfaces. We use generic Android or iOS devices so that no specialized equipment is necessary. Soundcool can also be used directly with the mouse, without any mobile devices. Another option is gestural control with a Kinect (Yoo, Beak, and Lee 2011). The system can be downloaded from soundcool.org, which has a collection of links to free resources such as Audacity and other free applications, VST instruments and effects, sounds, etc. The Soundcool mobile app to control modules over Wi-Fi is also available for free. Anyone can use Soundcool with his or her own computer independently of any classroom.

Modules

Soundcool includes an extensive set of modules, a selection of which are shown in Figure 3. At the time of writing, audio modules encompass: Record (from any input device or from another module); Player (plays at an indicated speed with optional looping and reversing, Figure 3a); Feedback Delay (Figure 3b); Pan; Transposer; Pitch Shift; Audio Routing; Mixer with 8 inputs; Spectroscope and Oscilloscope modules (Figure 3c shows the latter) visualize audio signals in the frequency and time domains; Sample Player loads and plays up to ten audio samples in one module; Direct Input captures microphone or line-level input; Filter provides ten different filter modes (Figure 3d); Signal Generator creates different kinds of waves based on frequency, amplitude, or

Figure 3. Examples of Soundcool modules, which run on a laptop or desktop computer: Player (a), Feedback Delay (b), Oscilloscope (c), Filter (d), Signal Generator (e), and Keyboard (f). Each module has a corresponding touch screen interface in the Soundcool app. A running app can be linked to a Soundcool module by entering a port number into the module interface and the app.



ring modulation (Figure 3e); Sequencer automates control of the signal generator module; Envelope; and VST Host incorporates VST instruments and effects. For VST instruments, we provide a virtual Keyboard module (Figure 3f) to receive notes and controls from the Soundcool mobile app; however, a physical MIDI keyboard connected to the computer running Soundcool can also be used.

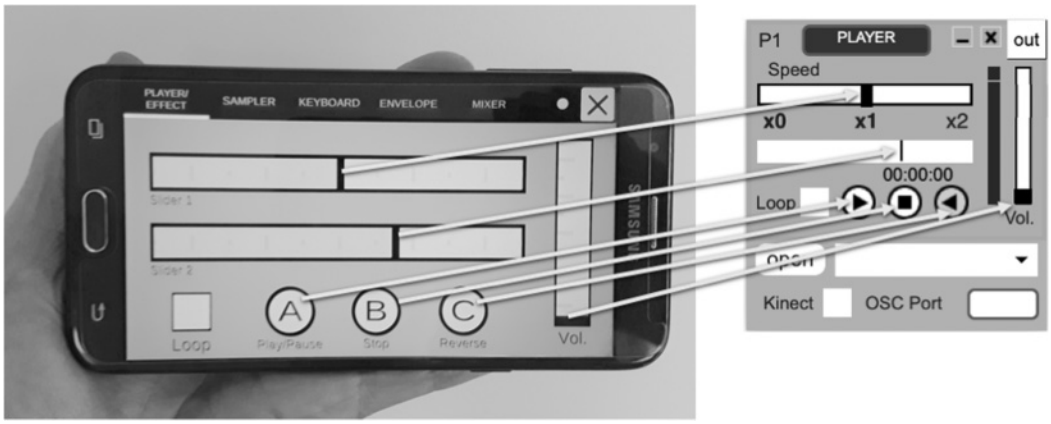
In general terms, modules have audio inputs and outputs, and outputs fan out to any number of inputs. To make a connection, the user clicks on an output, then presses an input button. Connections are indicated by color and text; for example, if Sample Player 1 (which has a red background) is connected to a Delay module, the input button of the Delay is colored red with the letters "S1" indicating the connection comes from Sample Player 1. The connected modules are also highlighted when the user hovers over an output button.

The choice of buttons as opposed to visual "wires" is mainly an implementation issue, but the absence of wires saves valuable screen real estate for module interfaces and their controls.

Implementation

Soundcool is implemented as a Max application, using the native objects `send` and `receive` (and their signal versions `send~` and `receive~`) for

Figure 4. OSC module control for smartphone or tablet.



patching. By implementing a patching system for Soundcool modules, we avoid exposing novice users to the full Max editing environment, and users can run Soundcool free of licensing fees. We are also developing a browser-based version of Soundcool using WebAudio.

Normally, a Soundcool system also includes networked mobile devices. Open Sound Control is used for communication between the host computer and any number of mobile devices (see Figure 4). The mobile devices must be configured manually with the IP address of the host computer where the modules are being run. Additionally, each Soundcool module is configured with a different receiving port number that must match the sending port of the controlling mobile device. This allows many devices to be used, each controlling a different module. Configuring each device with an IP address and port number is annoying at least, and we hope to use Zeroconf (Guttman 2001) to simplify connections in the future.

For the first Soundcool prototype we used TouchOSC (http://hexler.net/docs/touchosc), but we now offer a Soundcool-specific app written in Unity (unity.com) for Android and iOS. One of the features of our app is special handling of the keyboard to allow glissandi using sliding motions and to send extra note-off and all-notes-off messages when the last finger is removed from the keyboard, which helps to avoid "stuck notes" caused by lost OSC messages.

We now turn to example applications of Soundcool, ranging from classroom music education to network performance. We hope to illustrate the versatility and power of the Soundcool approach.

Soundcool in the Classroom

The original goal of Soundcool was to provide a hands-on creative environment for classroom music education. Soundcool is in use at different educational institutions in Spain, Portugal, Italy, and Romania through EU-funded Erasmus+ projects. Students learn fundamentals of computer music through collaborative and intuitive creation. Using Soundcool, students discuss creative ideas, experiment with sound production and manipulation in small groups, and perform entire pieces in larger group concerts (see www.youtube.com/c/SoundcoolProject).

Soundcool is designed for usability by young students and teachers who are not engineers or scientists. Soundcool modules are rather complete and self-contained units that provide an intuitive process (such as a mixer, sample player, etc.) with graphical controls, appropriate status display and level meters, OSC interface, and input/output patching controls. Wrapping so much functionality into each module allows for a minimal learning curve while still providing great creative potential, especially in education. The completeness of each module saves

users from building their own interfaces from low-level primitives, as seen in common programming languages.

In a typical primary school application, Soundcool provides a means of expression through a creative project in which students develop skills of listening and reflection (Sastre et al. 2013). Soundcool complements other modes of instruction through the body, voice, instruments, etc. Students begin with technical instruction on making sound and gradually learn the connection between control parameters and resulting sounds. Students are guided to create short pieces of a few minutes; for example, creating music to accompany a story. At first, students vocalize the sounds they imagine. Next, they may search the Internet for source sounds or explore VST plugin sounds. After configuring Soundcool modules, students rehearse and modify their pieces. The teacher can suggest sounds, rhythms, call-and-response interactions, etc., to further develop the composition. Then, performances are recorded, leading to further reflection. Discussion enables the joint construction of knowledge. Students tend to focus on emotion and feeling at first, but with practice develop understandings of sounds and their relationships.

An example is a story written by students at the elementary school CEIP Carmelo Ripoll in Valencia, Spain. In this work a girl tells a horror story, her live video image is blended with a scary image, and the rest of the students perform live music and ambient sounds, all using Soundcool (video processing is a 2018 addition to Soundcool modules, but beyond the scope of this article). The student production can be seen online (https://youtu.be/F3z9qoCnLiw [in Spanish]). To create this work, the teacher made the Soundcool configuration. Then the students selected all the sounds, rehearsed, and performed the sounds with smartphone and tablet controllers while one student narrated.

Soundcool and Algorithmic Control via OSC

One of the advantages of more general programming environments is the ability to implement sophisticated mappings between controllers and sound processes, as well as autonomous algorithmic control of parameters. Because Soundcool is intended for nonprogrammers, algorithmic mapping and control is not a built-in option. The Open Sound Control interfaces to Soundcool modules provide an interface for external programmatic control, however. This allows Soundcool to be viewed as a sophisticated modular synthesizer and allows the advanced user to focus on control aspects using any programming language.

As an example, the program in Figure 5, written in Serpent (Dannenberg 2002), works with several Soundcool modules to generate an interesting sound texture. The Soundcool modules consist of a variable-speed sample player with five short noise sounds connected to a feedback delay module. The program is a simple loop that runs about ten times per second. The program uses the function `send_afloat()` to send an OSC message containing one floating point number to a given address (the function definition is simple "glue" code and not included in this code excerpt). At each iteration, the program chooses, at random, one of eight actions to perform: the first five actions trigger one of five samples from the sample player, action 6 changes the delay time, action 7 changes the delay feedback level, and action 8 changes the player speed (transposing the apparent pitch of the samples). The time interval between events will approximate a negative exponential distribution, which is perceptually and musically interesting. By changing the SPEED parameter, the density of events increases from a sparse texture with great variety—due to changes in pitch and feedback delay—to a dense texture that could be compared to granular synthesis. Interested readers can hear the results online (www.cs.cmu.edu/~music/examples/soundcool-control.html). A similar approach could be taken using Python, Java, C, or even Pd or Max as the control program language. One could also receive OSC messages from controllers—for example allowing real-time, human control of the SPEED parameter from a touch interface while allowing the algorithm to generate detailed controls for Soundcool.

The reader might argue that resorting to an external program illustrates a weakness in Soundcool

Figure 5. A program to perform algorithmically using Soundcool. The program randomly triggers sounds and changes delay parameters to create a collage of noise sounds.

```
SPEED = 1 // if SPEED > 1, things happen more frequently
sounds = ["/4/push1", "/4/push2", "/4/push3", "/4/push4", "/4/push5"]

def sometimes(p) // return true with probability p * SPEED
    return random() > (1 - p * SPEED)

def run()
    while true
        for i = 0 to 5 // trigger sounds occasionally
            if sometimes(0.01)
                send_afloat(sounds[i], 1.0)
                time_sleep(0.01)
                send_afloat(sounds[i], 0.0)
                time_sleep(0.01)
        if sometimes(0.1) // maybe change delay time
            send_afloat("/1/fader2", random() * 1000)
        if sometimes(0.1) // maybe change delay feedback
            send_afloat("/1/fader1", random() * 1)
        if sometimes(0.05) // maybe change playback speed
            send_afloat("/4/fader1", random())
        time_sleep(0.1)
```

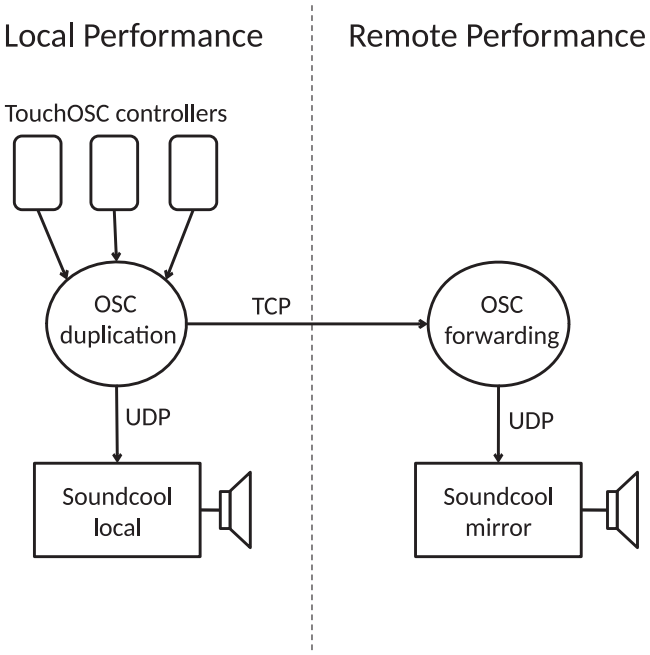
as a general-purpose computer music system. If automation and algorithmic control are essential for computer music, then perhaps they should be integral to any computer music system. On the other hand, for users (as opposed to language designers), the ability to control Soundcool with an already familiar language might be a feature.

Soundcool in Telematic Performances

Soundcool has been used for telematics performance (http://globalnetorchestra.blogs.upv.es). Because Soundcool control is normally through mobile devices, the control information is inherently a stream of OSC messages. For a telematic performance, we duplicated the local OSC message stream, transmitted it to a remote performance site, and fed the messages to a remote duplicate of the local Soundcool configuration. As a result, the remote site had a "mirror" of the local Soundcool performance, albeit with some network delay and timing jitter. We dealt with that in advance by designing sound textures that did not have precise timing requirements.

The configuration is illustrated in Figure 6. The OSC Duplication program is both an OSC client and server, receiving from controllers and forwarding to the local Soundcool. The controllers are set up with different port numbers to connect them to OSC Duplication rather than directly to the local Soundcool program. The OSC Duplication program uses a TCP connection to transmit to the OSC Forwarding program at the remote site. The TCP protocol is used here because the UDP protocol with its "best effort" delivery is prone to losing messages, whereas TCP will automatically detect and retransmit dropped messages (at the cost of holding up delivery of subsequent messages until retransmission is successful). The OSC Forwarding program is also an OSC client. It sends arriving messages locally over UDP as expected by the remote Soundcool Mirror process. The OSC duplication and forwarding programs, written in Serpent, total only 90 lines of code and are available from the authors.

Figure 6. Telematic performance with Soundcool: OSC messages are duplicated and relayed to the remote site, creating a "mirror" performance.



Our point here is not that Soundcool is the best tool for telematic performance. Indeed, many approaches and systems have been created, often exploring particular ideas about the role of time, distance, synchronization, and communication in musical interaction (Mills 2019). Instead, our point is that such a simple tool can be extended relatively easily to serve a purpose that was not anticipated by the original designers. Moreover, the design of Soundcool allowed us to completely separate composition and performance aspects (in Soundcool) from the network transmission aspects, which greatly simplified the project. During the Covid-19 pandemic, we have used Soundcool to connect isolated but collaborating performers in as many as eight cities (https://youtu.be/rXQ73PWxzSk).

Evaluation

Computer music systems, like other complex languages and systems, have no simple basis for evaluation. However, we can attempt to answer a few questions through surveys and field tests. We will address the following questions:

1. Does Soundcool have potential for adoption in the classroom?
2. Is Soundcool powerful and simple enough for use by young students?
3. Is Soundcool general enough to be interesting to professionals?

We offer a user study, adoption by schools, and use by professional performers as evidence that Soundcool meets a range of needs.

To answer the first question, Soundcool and music technology was introduced to a group of 66 teachers in a workshop setting (Murillo, Riaño-Galán, and Berbel-Gómez 2018). After working with Soundcool, a number of positive outcomes were reported:

1. Teachers are highly prone to use computers in education (86%) even though only 58% reported prior computing experience.
2. Teachers feel more positive about Information and Communication Technology (73%).
3. Teachers believe Soundcool will increase the generation of ideas by students (91%).

This study shows that teachers can learn and embrace Soundcool for music education.

To answer the second question, "Is Soundcool powerful and simple enough for children?" we have extensive experience with Soundcool in schools. Although outcomes are hard to measure objectively, and controlled comparative studies are even more difficult, we can subjectively see successful student engagement, creativity, listening, performance, and mastery of digital media. In the majority of cases, this experience was completely lacking before the introduction of Soundcool to the curriculum. Soundcool has been used in 25 educational centers, and student creations include operas, plays, silent film scores, and audio/visual storytelling. Many productions can be seen online (www.youtube.com/c/SoundcoolProject). Some of these projects are available with instruction manuals in both English and Spanish and video examples for teachers (http://soundcool.org/en/projects).

Finally, a growing list of works and performances suggests that Soundcool has much to offer, even

to professionals. The opera *La Mare dels Peixos* [*The Mother of Fishes*], for six soloists, children's chorus, orchestra, and live Soundcool electronics has been performed in Spain, Mexico, and the United States (http://themotheroffishes.com). A video excerpt of the opera is available at https://www.mitpressjournals.org/doi/suppl/10.1162/COMJ_a_00534.

For each performance, children or young adults learn Soundcool, develop sounds and music, and perform live in a professional setting (http://bit.ly/tmof-audacity). Soundcool has also been used in a growing list of other contemporary music performances. A playlist can be found online (http://bit.ly/soundcool-youtube). In general, the attraction of Soundcool is simplicity and support for multiple effects and sample playback, all of which can be quickly configured, adjusted, and controlled. For example, *Chapitres* for wind symphony, narrator and Soundcool, uses Soundcool to capture live sound fragments from a voice and vibraphone and process them with two separate granular synthesis effects (Tom Erbe's Bubbler, www.soundhack.com, and INA-GRM tools, inagrm.com). Eighteen modules including three VST host modules provide a wide range of processing possibilities and parameters (youtu.be/f_Wt3fKi82E).

Summary

We have presented Soundcool, a system for innovative music education based on collaborative creation using mobile devices. Soundcool is designed to fill the gap between off-the-shelf, ready-to-use applications and programming-oriented software development platforms. Soundcool presents a minimal learning curve while still providing high creative potential, especially in education. We have described the design and rationale of Soundcool including the Soundcool app used for multitouch control and collaborative performance.

We have showcased three different scenarios of use as examples of how Soundcool is powerful but still sufficiently general for interesting and even unexpected applications. The first scenario describes music education in classroom settings. The second scenario concerns the algorithmic control of Soundcool from other pieces of software via the OSC interface implemented in every Soundcool module. The third scenario describes the use of Soundcool in a telematic performance.

We use surveys and experience to support our claims that (1) Soundcool offers an interesting basis for expanding music teaching in primary and secondary education, (2) Soundcool is simple and powerful enough for use by children, and (3) Soundcool is also sufficiently advanced and general enough for professional composers and musicians. In the future, we hope to accelerate the adoption of Soundcool by offering a browser-based implementation and a cloud-based system for music creation and sharing.

Conclusions

We believe that Soundcool is interesting in at least two ways: As an approach to music education, Soundcool leverages modern technology to motivate collaborative music creation and an openness toward nontraditional sound and music. As a system design, Soundcool runs counter to efforts to create "all-in-one" systems that offer signal processing, control, interfaces, programmability, and more. Soundcool makes the case that certain features can combine to form a powerful and flexible system for computer music. From experience, we can identify three essential features, and we encourage designers to consider the following as important enablers of creative practice.

First, high-level modules for "standard" processing, such as mixers, sample players, and filters are so widely used that there is no sense in having users build their own. Modules include simple, real-time graphical interfaces so that modules are immediately useable. Although systems that are more sophisticated tend to separate the audio interconnection interface from the control interface, there is value in putting all of a module's information and interactions within a single visual representation.

Second, audio plugins vastly expand the availability of interesting and sophisticated audio signal processors and generators. The VST host module in Soundcool allows users to incorporate complete

synthesizers, high-quality reverberation effects, and a wide variety of signal processors, without complicating the basic framework.

Finally, separating signal processing and control opens many control possibilities, including multiple touch surfaces, new control devices, telematic control, and algorithmic control using virtually any programming language. Close coupling between control and audio and even sample-accurate control in computer music languages is a good thing, but a more distributed and modular approach to control brings both simplicity and flexibility, hence more power and generality. Assuming audio modules with substantial functionality, it is important that module parameters be ready-to-use, avoiding the need to manually connect each parameter to some control source. For example, Soundcool users merely enter a port number for OSC messages, and modules become fully connected to the touch controls of the Soundcool app.

Although Soundcool was created for primary and secondary education, it has found a variety of uses owing to its combination of power and generality. We hope that others will try Soundcool. We also hope that some of the lessons we have learned will help systems designers to create systems that are both more flexible and usable in the future.

Acknowledgments

This work has been supported by the Generalitat Valenciana (Spain) (grants GJIDI/2018/A/169 and AICO/2015/120) and the Daniel and Nina Carasso Foundation (grant 16-AC-2016). We would also like to thank Carnegie Mellon University, the Heinz Endowments, the Greater Pittsburgh Arts Council, and our students for their support of Soundcool, *The Mother of Fishes* opera, and participation in the telematic performance described here.

References

Baumann, A., et al. 2011. "Enhancing STEM Classes Using Weave: A Collaborative Web-Based Visualization Environment." In *Proceedings of the Integrated STEM Education Conference*, Paper 2A.

Clough, G., et al. 2008. "Informal Learning with PDAs and Smartphones." *Journal of Computer Assisted Learning* 24(5):359–371.

Dannenberg, R. B. 2002. "A Language for Interactive Audio Applications." In *Proceedings of the International Computer Music Conference*, pp. 509–515.

Dannenberg, R. B. 2018. "Languages for Computer Music." *Frontiers in Digital Humanities*. Available online at doi.org/10.3389/fdigh.2018.00026. Accessed June 2020.

Freeman, J., et al. 2015. "EarSketch: A STEAM Approach to Broadening Participation in Computer Science Principles." In *Proceedings of the International Conference on Research in Equity and Sustained Participation in Engineering, Computing, and Technology.* Available online at doi.org/10.1109/RESPECT.2015.7296511. Accessed June 2020.

Guttman, E. 2001. "Autoconfiguration for IP Networking: Enabling Local Communication." *IEEE Internet Computing* 5(3):81–86.

Lloret Romero, Nuria. 2007. "Study of Human-Technology Interaction in e-Learning Platforms Design." *International Journal of Technology, Knowledge, and Society: Annual Review* 2(6):21–28.

Manzo, V. J. 2015. *Foundations of Music Technology*. 1st ed. Oxford: Oxford University Press.

Mills, Roger. 2019. *Tele-Improvisation: Intercultural Interaction in the Online Global Music Jam Session.* Berlin: Springer.

Murillo, A., M.-E. Riaño-Galán, and N. Berbel-Gómez. 2018. "Perception of the Use of 'Soundcool' as a Proposal for Intervention in the Creation of Sound and in the Development of Teaching Competences: An Exploratory Study on Pre-Service Teacher Education" [In Spanish, with English abstract]. *Psychology, Society, and Education* 10(1):127–146.

Newell, A. 1983. "Intellectual Issues in the History of Artificial Intelligence." In F. Machlup and U. Mansfield, eds. *Study of Information: Interdisicplinary Messages.* Hoboken, New Jersey: Wiley, pp. 187–227.

Sánchez, J., A. Salinas, and M. Sáenz. 2007. "Mobile Game-Based Methodology for Science Learning." In A. Jacko, ed. *Human–Computer Interaction: HCI Applications and Services.* Berlin: Springer, pp. 322–331.

Sastre, J., et al. 2013. "New Technologies for Music Education." In *Proceedings of the International Conference on E-Learning and E-Technologies in Education*, pp. 149–154.

Sastre, J., et al. 2015. "Soundcool: New Technologies for Music Education." In *Proceedings of the International Conference of Education, Research and Innovation,* pp. 5974–5982.
Savage, J. 2007. "Reconstructing Music Education through ICT." *Research in Education* 78(1):65–77.
Shapiro, R., et al. 2017. "Tangible Distributed Computer Music for Youth." *Computer Music Journal* 41(2):52–68.
Yoo, M.-J., J.-W. Beak, and I.-K. Lee. 2011. "Creating Musical Expression Using Kinect." In *Proceedings of the International Conference on New Interfaces for Musical Expression,* pp. 324–325.