Wubify.me

Yanni Coroneos

Massachusetts Institute of Technology, Electrical Engineering and Computer Science

Programmer

Corey Walsh

Massachusetts Institute of Technology, Electrical Engineering and Computer Science

Programmer

Abstract

In this paper, we describe a continuation in the modern trend of audio synthesis based on the concept of generators, filters, and amplifiers. By combining the achievements of previous products and taking into account the needs of the user, we propose wubify.me. We assert that users at every level of expertise can meaningfully benefit from the extended feature set, and we especially believe that our shareable user-contributed block library can elevate the experience of audio synthesis to a social activity.

Introduction

Music today is often made with synthesizers - either physical devices that string synths together via audio cables, or computer programs that do so with bits and bytes. In either case, there is a large learning curve, and the software or hardware can be inhibitively expensive. Wubify.me attempts to mitigate both of these obstacles - increasing accessibility of such tools to the general public.

Images removed due to copyright restrictions. Please see Works Cited on the last page to view images.

Left: Example physical controller [3]. Right: Daft Punk's Pyramid [5]

Background

The need for wubify.me was realized when a hobbyist electronic music composer expressed interest in a visual composition system. We decided to investigate the topic further.

Mixers and Synths

The first step was to look at the physical tools that countless real musicians use every day: mixers, synths, and other related equipment. We noticed a general trend: in the abstract sense, each connected component could be described as a **filter**, a **generator**, or an **amplifier**. Audio cables act as highways, linking together a large and flexible audio generation pipeline.

SuperCollider

SuperCollider is the current leader in the field of algorithmic music generation software. According to their website[4], Supercollider is "a programming language for real time audio synthesis and algorithmic composition." What this approximately means, is that it provides low level access to the same set of concepts that we encountered with physical mixers and synthesizers, in a digital environment. With these tools, artists can interact with the basic concepts of filters, generators, and amplifiers, through a computer command line interface rather than the physical counterparts.

Not only does SuperCollider expose the functionality found in real hardware for free, it also does it in a way that provides a high level of mathematical rigor. Physical devices are unable to afford the same correctness guarantees, as they are exposed to the imperfections of real-world interactions such as signal

degradation. This mathematical backing, and the dynamic nature of software, allows SuperCollider models to be easily and quickly abstracted from basic building blocks (such as sine wave oscillators) to complex generators or filters composed of many such building blocks.

While SuperCollider provides a great platform for algorithmic music generation, it's Achilles' heel lies in the fact that it is challenging to use. It's primary interface is through textual computer code - a great format for conveyance of intention, but not as great for human users.

Concept

We set ourselves on a solution that keeps the flexibility and power of SuperCollider along with the ease of knobs and buttons. This visual system exposes all of the basic building blocks of synthesized music to the user. Unlike the old solutions, though, it will be easy to use, require little background knowledge, and run completely on the web.

Every operation can be done with visual blocks and the connections between them. In a workflow reminiscent of SuperCollider, the user can arrange generators and filters in order to transform an input sound. Wires between the blocks will carry the same signals as wires between analog mixers. By leveraging powerful technology, we can move all aspects of electronic synthesis to the digital world and simplify it at the same time.

The digital domain also opens doors to new possibilities that aren't possible with the spatial limitations of physical devices. Wubify.me aims to combine of the best aspects of SuperCollider and professional mixers. Its level of abstraction allows the user to visually manipulate the signal chain like a mixer, but it also allows for the creation of original content.

Since our application runs entirely on the web, we can also allow users to work collaboratively when creating sounds. This feature can play nicely with another one that allows users to submit their finished synths to a user block library that everyone can access and contribute to. Of course, each user can also have his own private block library so that others can't change their blocks. This kind of functionality is not possible with a single physical interface. The proposed concept opens the doors to easy abstraction from low level generators, filters, and amplifiers to higher level objects, similar to physical mixers in presentation and function.

When considered together, these features allow for augmenting the topology of a digital mixer - or similar algorithmic devices - by giving it a narrative and history.

Discussion

Sonic Functions as Objects

Our representation of sound as discrete blocks is a continuation of recent changes to the overall paradigm of composition. Instead of notes, the basic units become generators, filters, and amplifiers. Generators produce basic sounds which run through filters and the output of those gets run through an amplifier. Because of this, the topology of these objects suddenly becomes the metric of quality, and not so much the actual individual sounds themselves. Interest has been shifted to the chain of signal operations and not the signal itself because the input signal can change but the function performed on it will remain the same.

A Narrative With History

The user block library simultaneously serves as a database (in all senses of the word) and as a medium of communication. Much like how street art is often added to, users can augment already existing blocks and, in this way, leave their own mark on the narrative. Once submitted, synth blocks could be reused as if they were a single block. If work is unsaved and public, there is a chance that others may find it and modify it fit their needs or style. This allows the library of publically available synth blocks to adapt to the temporal needs of the users. By making synthesis more accessible, a wider range of people can contribute to the narrative. There is also power in numbers. Increasing users crowdsources the problem of designing interesting synths and we hope this can result in novel sounds.

Real-Time Collaboration

With the introduction of real-time collaboration on public synth blocks, there is a level of fluidity that is introduced. Suddenly, many users can simultaneously work on a single synth block at the same time. There is no single author. This leads to a workflow similar to that of a Google Doc with multiple collaborators or even a Wikipedia page. Feedback is instant and creativity abounds.

Implementation

As a proof of concept, we have begun an implementation of the above described system, which we will discuss throughout this section.

Enabling Technologies

Our proof-of-concept application would not be possible without the help of a long list of enabling technologies. Most notably:

Flocking.js[2] - Flocking is a SuperCollider inspired JavaScript library that provides a similar interface and operation. It has the benefit of being executable in the browser - greatly simplifying the process of delivering audio output to our users

JointJS/Rappid[1] - “A complete diagramming toolkit.” [1] JointJS is a JavaScript library that provides support for visualization and interaction with graphs. Rappid provides a set of higher level features that greatly sped up the development process of our proof-of-concept. While JointJS has a free and open source MPL 2.0 license, Rappid is a commercial product that was provided free of charge for our academic use.

Node.js - A leading, open source web-server platform written on top of Chrome’s high performance V8 JavaScript runtime. Node provides a platform for both our file serving needs and our application’s runtime requirements.

Development

The implementation of this project decomposes into a few parts:

1. **Flocking.js Representation**
2. **Graph Representation**
3. **User Interface - Shared block library, real time collaboration**
4. **Web Serving**
5. **Persistence**

The primary ‘brains’ of the technology is the algorithm that converts the graph representation into the valid JSON object that Flocking.js requires. This algorithm traverses the graph, at each node it generates a corresponding Ugen descriptor, populated with default values and user specified options. Finally, it sews the fragments into a cloth that represents the network of data commutations.

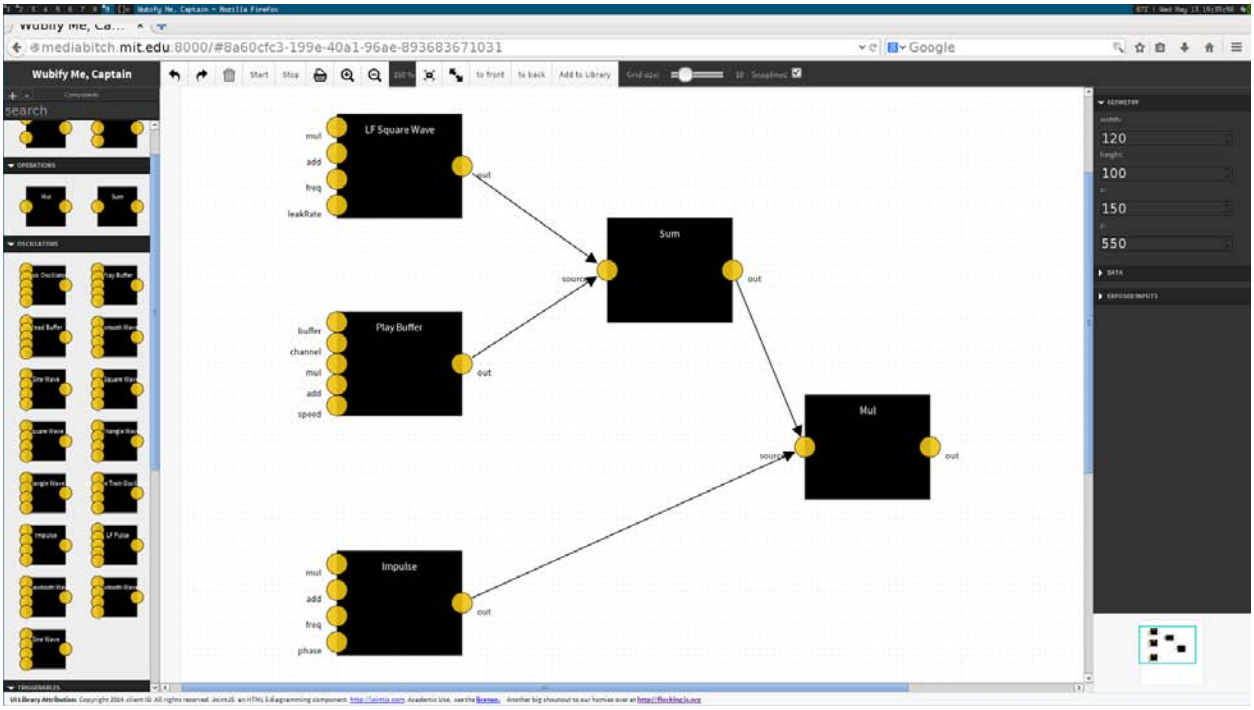

Fig 1. Graph representation of the signal chain

Rappid.js and Joint.js provide the basis for our demo user interface. Most other levels in of implementation - such as web serving and persistence - rely on fairly standard and freely available tools, discussed further in the Enabling Technologies section.

Journey Map

Wubify.me has a straightforward UI. When a user logs in, they are presented with a new synth block creation interface. The sidebar provides a set of base level unit generators (ugens) which form the basis in terms of filters, a generators, and amplifiers. Users can build their own synth blocks from these ugens, as well from the other contributed user ugens that appear on the sidebar.

The user drags a few ugen blocks into the empty space and notices that they have input and output ports. By dragging arrows from outputs to inputs, the user can cascade the ugens. Mathematically, this is equivalent to cascading transfer functions - each function commutating the input data in some well defined manner.

Each ugen may vary but common inputs are “add,” “mul,” “freq,” and “phase.” Connecting an output to each of these ports either adds or multiplies the converging signals, or modifies the base frequency or phase angle of the target ugen block, respectively. From there the user composes ugens and then hears the result by clicking the play button.

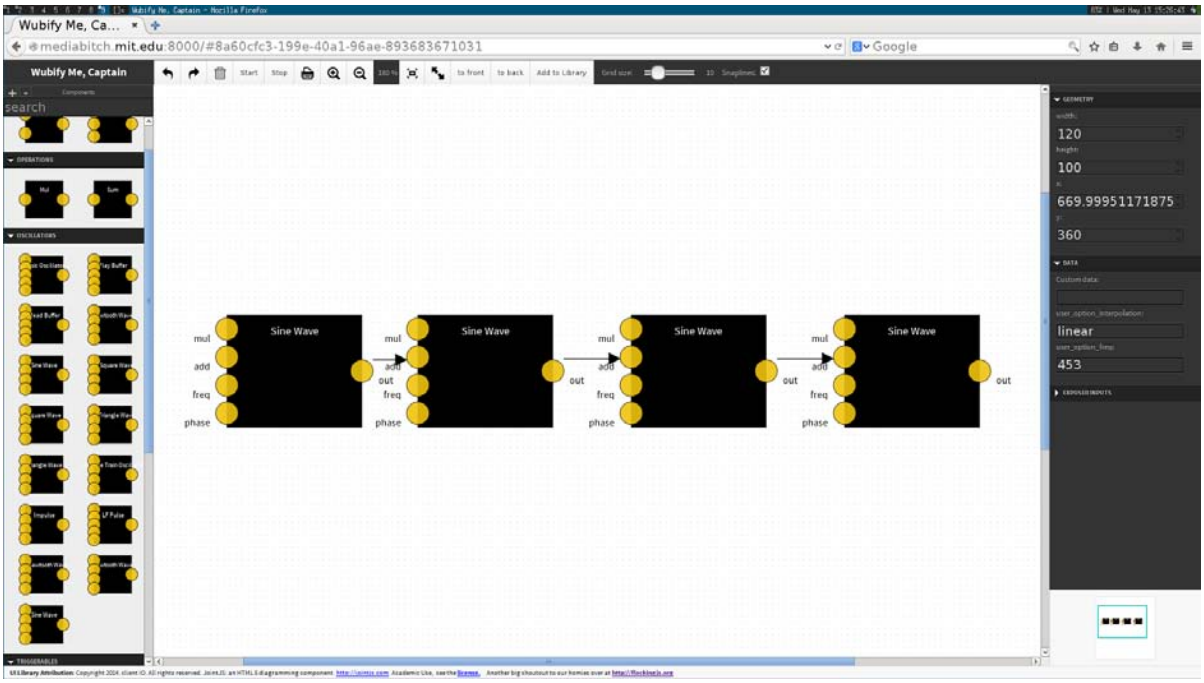

Fig 2. A chain of ugens that results in a beat frequency

If desired, the current block state can be saved into a new synth block in the user block library by clicking the save button.

Future Directions

Our proof of concept application provides the basics, but we believe that this project could be taken further.

A built in sequencer would allow for more natural coordination of multiple generators. This would give the user a similar manner of control to programs such as Garageband or Audacity. The ability to string synth blocks together in time is certainly a powerful one and it definitely needs to happen. Currently, a similar effect can be painfully achieved with multiple delay blocks and triggers, but this only highlights the need for a sequencer.

While our implementation is built on Flocking.js, only relatively minor changes would be required to make it work with SuperCollider. Aside from the availability of basic a few basic synth blocks, the UI would be virtually unchanged; only a new graph parser would be required to generate valid SuperCollider code.

Since SuperCollider cannot run directly in the web browser, the downside of this method is that users would have to accept networking lag due to the time required to communicate with SuperCollider servers, or run their own local server to avoid the network. The former is easier from the user's point of view, but may not be responsive enough for professionals, conversely the latter requires more configuration and know-how, but provides faster response times for real-time music creation. One benefit of this approach is that users could choose options depending on their experience level or requirements.

Our abstractions of signal transformations into blocks can be applied to other fields. For example, a lot of video processing also relies on a signal chain of filters and amplifiers so perhaps we could tailor our existing implementation for that type of use. Ugens would emit a video and the filters would be

two-dimensional instead of one-dimensional. The prospect of having real-time collaboration on a video project seems great.

Conclusion

Wubify.me aims to bring synthesizers and mixers into the 21st century. By combining powerful technology with innovative social additions, we hope that electronic music synthesis can become more fun and accessible to everyone. SuperCollider gives us a base to build from and Rappid.js gives us the means to visually represent sound. A working implementation is really just the beginning; the rest comes from you.

Works Cited

[1] http://jointjs.com/about-rappid

[2] http://flockingjs.org/

[3] http://ecx.images-amazon.com/images/I/71pG5fR-4wL._SL1461_.jpg

[4] http://supercollider.github.io/

[5] https://digitaldj.files.wordpress.com/2008/07/daft_pyramid.jpg

CMS.633 / CMS.833 Digital Humanities

Spring 2015