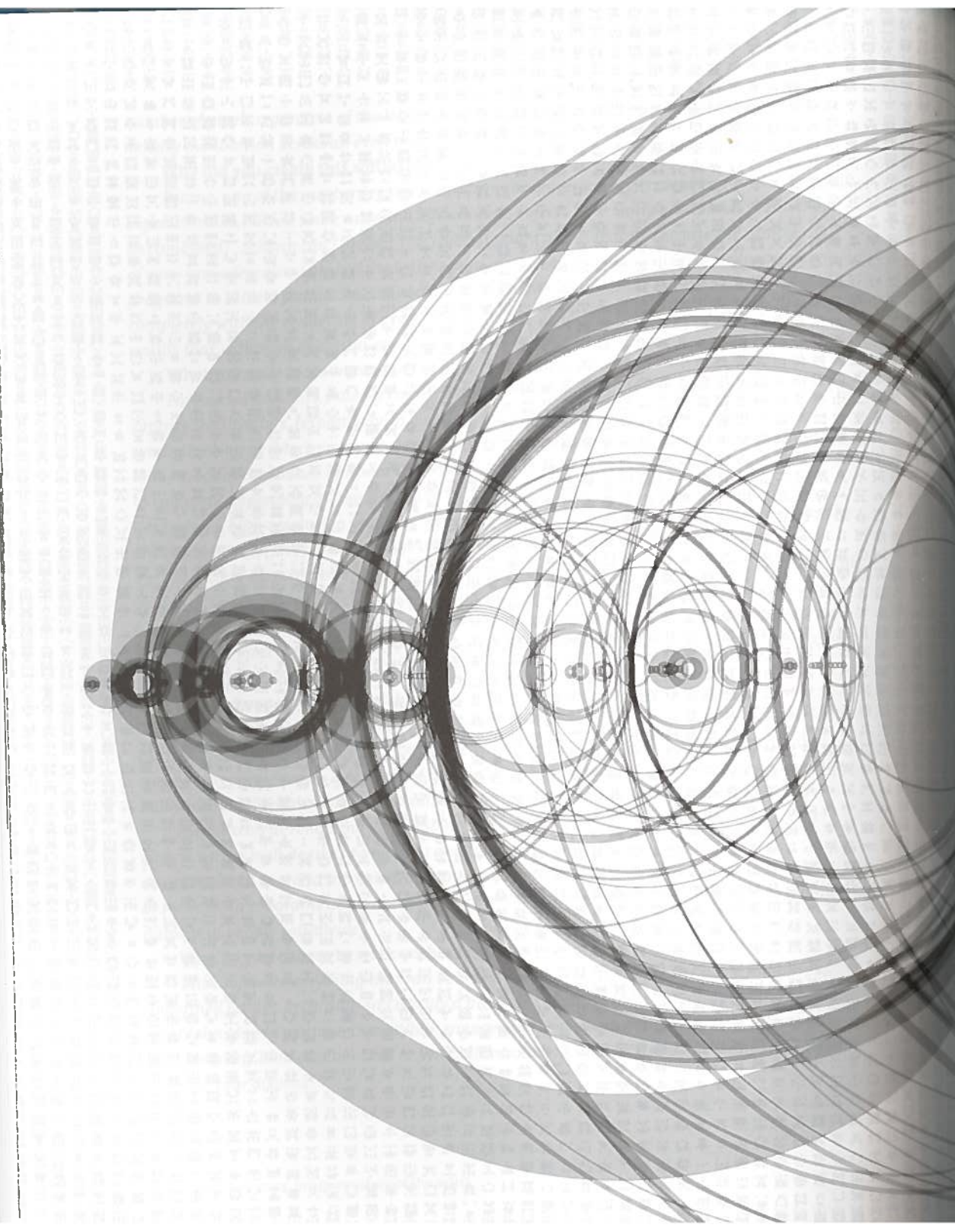
Processing: a programming handbook for visual designers and artists

Casey Reas
Ben Fry

The MIT Press
Cambridge, Massachusetts
London, England



Shape of Song *(Interview with Martin Wattenberg)*

Creator	Martin Wattenberg
Year	2002
Medium	Software, Prints
Software	Java
URL	www.turbulence.org/Works/song

What is *Shape of Song*?

The Shape of Song *is an attempt to answer the seemingly paradoxical question "What does music look like?" The custom software in this work draws musical patterns in the form of translucent arches, allowing viewers to literally see the shape of a composition.*

One of the satisfying aspects of the visualization, to me, is that different musical styles translate to characteristic, distinct visual styles. Folk songs yield simple, repetitive arrangements. Classical pieces have an almost mathematically precise visual structure. Jazz translates to my favorite diagrams, for they often start out with extremely regular patterns and then devolve into something close to chaos.

The work itself has existed in many different forms. It began as a program that I could only run myself. Later I turned it into a Web-based project that let viewers upload their own music. Watching people upload works was fascinating: viewers often tried "extreme" music (the most atonal, the noisiest, the silliest top-40 tunes) to stretch the visualization. Many people are startled when they look at visualizations of "low-culture" music, such as a Led Zeppelin song, because the diagrams are so complex. So the artwork is a good anti-snobbery machine.

Finally, I created prints of some of my favorite diagrams. Most of my work has been purely screen-based, so it was a bit of adventure to work on paper. The level of detail provided by print is wonderful. To see all the different scales of structure at once is beautiful, I think.

I'm currently working on a more dynamic version that will include temporal aspects. My goal is to weave together more closely the spatial rhythms with the actual rhythm of the underlying music.

Why did you create *Shape of Song*?

This project was a personal exploration of the nature of music, balance, and the translation between eye and ear. Music visualization has been a subject of interest for centuries, which is one of the appeals of working on it: you have the sense that you are part of history.

I wanted to understand some of the symmetries found in music. Much of music visualization is aimed at literal translations of notes and rhythms into color and animation. Something more abstract appealed to me, and I pursued a representation of the overall musical form instead.

Although the images created in The Shape of Song *are far from a literal translation of the music, the arc-based diagrams came closer than anything else to expressing the mystery and beauty I feel when listening to the underlying compositions.*

Visualization of Vivaldi's Concerto No. 3 in F major, Op. 8, RV 293, "L'autunno." Image courtesy of the artist.

Why did you write your own software tools?

I had to; they didn't exist yet!

A more nuanced answer includes the fact that I actually used a great deal of existing work when writing the code. The method I used to analyze the music is a standard structure in computer science known as a suffix tree. (Suffix trees work by turning sequences into trees and are traditionally used for rapid searching of text. I am always happy when a piece of computer science, designed for mundane purposes, turns out to be useful in an artwork.) During the course of the project I also used a variety of libraries: some provided by Sun, the developers of the Java programming language, for graphics; and some written by others for writing graphics files and reading scores in the "MIDI" music format.

The use of the MIDI format is a quirk of the piece: more common formats, such as MP3, are harder for my algorithms to handle. (You can think of MIDI as analogous to vector graphics, while MP3s are like JPEGs; if you're trying to find simple patterns like squares or circles, it's far easier with a vector format!) At the time I first started working on the piece, MIDI files were extremely common, but they are becoming more and more rare. That raises some questions about the longevity of the piece, but perhaps by the time MIDI is obsolete someone will develop a reliable algorithm for translating MP3s into musical scores.

To sum up, while the goal of my code was original, most of the computer's time is spent in algorithms or code developed by others.

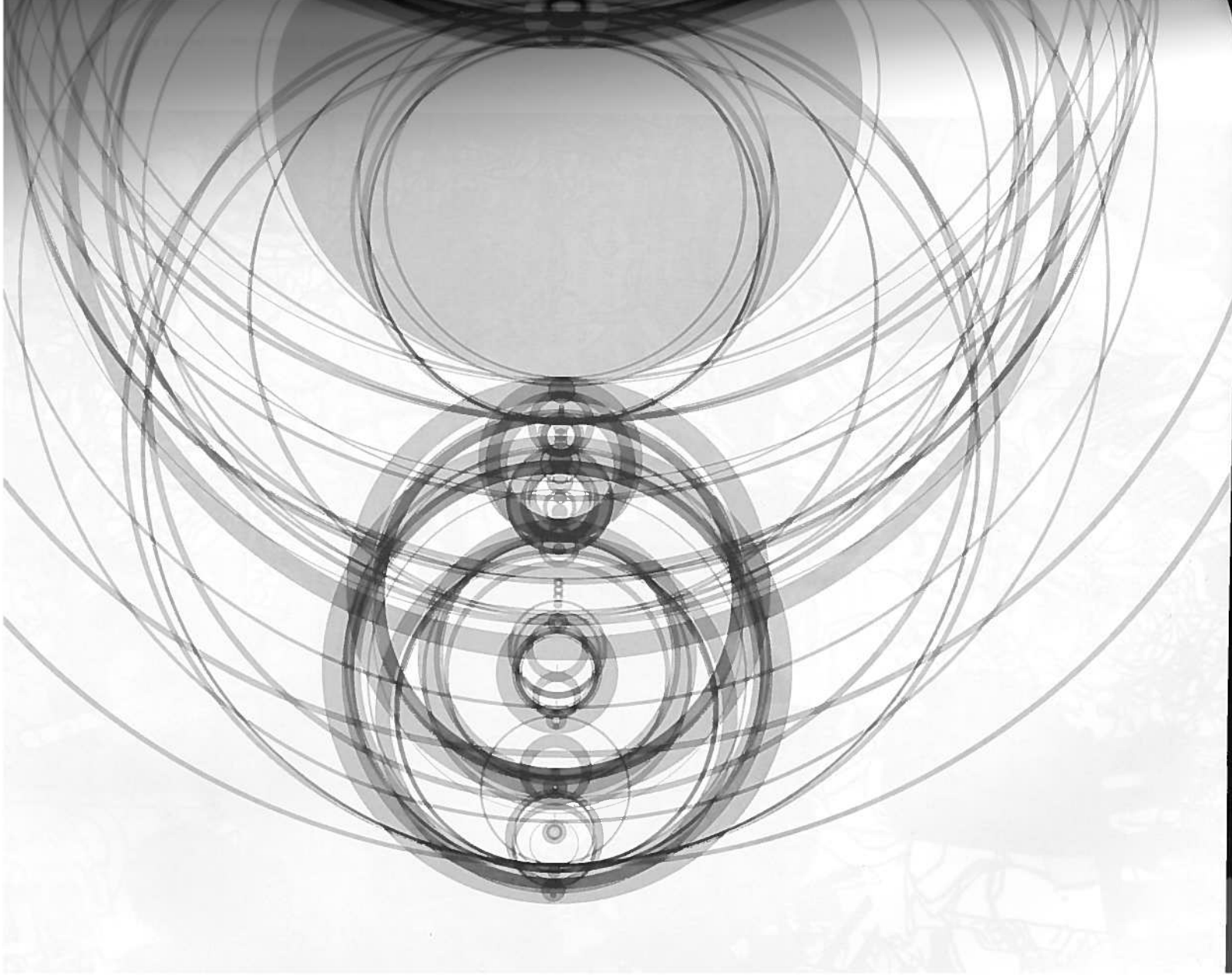
Why do you choose to work with software?

Software is the best way I've found to express myself. When I work in other media, the results somehow always seem worse in reality than in my head. The software I create, however, has a magical quality: it ends up being better than what I originally imagined.

To put it more metaphorically: when I create art, I feel like I am in conversation with the artwork. If I sketch or write, it's like talking to a caustic debater, exposing all the flaws in my thinking. Valuable, perhaps, but also discouraging! When I write programs, I have the opposite feeling: that I am talking with a sympathetic and brilliant partner who helps me organize my thoughts and points out connections I hadn't seen myself.

A second attraction of software art is that it is a new, growing field. There is a kind of energy associated with beginnings that I love. Each new piece seems like it's pointing out new directions, and there's a feeling that you're in a group of settlers on a frontier. As with the American frontier, some people settle down and found new cities, some people keep finding new paths, and some discover gold mines.

Visualization of Vivaldi's Concerto No. 3 in F major, Op. 8, RV 293, "Lautunno." Image courtesy of the artist.




will be delt
by Russia and
I bet

Listening Post *(Interview with Mark Hansen)*

Creators	*Mark Hansen and Ben Rubin*
Year	*2001–2002*
Medium	*Installation*
Software	*Perl, C, Max/MSP, C++, sh/tcsh, R*
URL	*www.earstudio.com/projects/listeningPost.html*

What is *Listening Post*?

Listening Post *is an art installation that culls text fragments in real time from unrestricted Internet chat rooms, bulletin boards, and other public forums. The texts are read (or sung) by a voice synthesizer, and simultaneously displayed across a suspended grid of 231 small electronic screens (11 rows and 21 columns).* Listening Post *cycles through a series of seven movements (or scenes) each with a different arrangement of visual, aural, and musical elements and each with its own data-processing logic.*

Why did you create *Listening Post*?

Ben and I met in November of 1999 at an event sponsored by Lucent Technologies (my former employer) and the Brooklyn Academy of Music. For our first project, we created a "sonification" of the browsing activity across a large, corporate website. Sonification refers to the use of sound to convey information about, or to uncover patterns in, data; it seemed like a reasonable place to start for a sound artist (Ben) and a statistician (me). We spent several weeks creating an algorithm that translated patterns of user requests into music. The mapping was pretty direct, differentiating traffic through major areas within the site (defined by a handful of top-level directories) and the depth to which people wandered (again, measured in terms of the site's directory structure). Unfortunately, it was tough to get anyone to take notice; even the site's content providers were hard-pressed to find a reason to listen. After a month or so we decided that perhaps navigation statistics (a by-product of the actions people take on the Web) were less interesting than the substance of their online transactions, the content being exchanged. We also agreed that the act of Web browsing wasn't very "expressive" in the sense that our only glimpse of the users came from patterns of clicks, lengths of visits, and the circle of pages they requested. These considerations led us to online forums like chat and bulletin boards. (Of course, this was early 2000; had we started our collaboration today, blogs or YouTube.com *or even* MySpace.com *might have been more natural next steps.)*

In retrospect, it was pretty easy to create a data stream from these forums, sampling posts from various places around the Web. Doing something with it, representing it in some way, responding to its natural rhythms or cycles, proved to be much harder. Text as a kind of data is difficult to describe (or model) mathematically. To make matters worse, online forums are notoriously bad in terms of spelling and grammar and many of the other bread-and-butter assumptions underlying techniques for statistical natural language processing. However, I think our interest in online forums went beyond summarizing or distilling their content (reducing the stream to a ticker of popular words or topics). Instead, we wanted to capture the moments of human connection; and in most cases these refused to be mathematized. Early in our process,

Listening Post, 2001–2002. Images courtesy of the artists.

we decided to let the data speak for itself in some sense, creating scenes that organized (or, formally, clustered) and subset the content in simple, legible ways.

Building on our experience with the Web sonification project, our first experiments with chat were sound pieces: A text-to-speech (TTS) engine gave the data a voice (or voices, as there might be up to four speaking at one time), and we created different data-driven compositional strategies to produce a supporting score. As we listened, however, we found ourselves constantly referring to a text display I hacked together to monitor the data collection. While we were led to this simple visual device to help make up for deficiencies in the TTS program ("Did someone really type that?"), it soon became an important creative component. This visual element evolved from a projection with four lines of text (at a live performance at the Kitchen in 2000), to a 10 by 11 suspended flat grid of VFDs, vacuum fluorescent displays (the first installation of Listening Post at the Brooklyn Academy of Music in 2001), and finally to the arched grid of 231 VFDs (first exhibited in 2002 at the Whitney Museum of American Art). Listening Post's visual expansion was accompanied by the introduction of a new TTS engine that let us literally fill the room with voices (as many as a hundred at one time).

What software tools were used?

The behavior of each individual VFD is ultimately directed by an onboard microcontroller running a custom C program written primarily by Will Pickering at Parallel Development. The screens are then divided into 7 groups of 33 (each an 11 by 3 subset of the entire grid) and are fed messages by 7 servers that listen for commands to display text along columns or on particular screens. The basic screen server is written in Perl. One layer up, the arched VFD grid is choreographed via a series of scene programs, again written in Perl.

The audio portion of Listening Post involves dynamic musical composition orchestrated by Max/MSP; messages are sent to Max from the scene programs via the Open Sound Control (OSC) protocol. It's worth noting that the programming interfaces for the audio and visual portions of Listening Post are very different; while Max is a visual programming environment, meaning that Ben directs output from one sound component to another by making connections in a "patch," I hack about in an Emacs window combining subroutines from a main scene module. The last major piece of software directly involved in the presentation of Listening Post is the TTS engine. Like Max, the TTS engine receives messages from the scene programs; unlike with Max, however, we had to write a custom C++ wrapper to handle the network communication. Aside from Max and the TTS engine, there are also other, perhaps less obvious, software components hidden in the system. The installation itself involves eight speakers and, during each scene, the voices and other musical elements move around the room. While Max handles much of this motion, a Yamaha Digital Mixing Engine (DME) is also used, which in turn requires a separate program for each of the scenes.

Finally, we made use of a slightly different set of software tools during scene development. At a practical level, each new scene consists of a Perl program orchestrating the visual elements and controlling the overall scene structure and a Max patch/DME program pair creating the scene-specific audio. (At this point, we treat the VFD grid and the TTS engine as fixed-output devices whose programming does not change with scene; they respond to a predetermined set of commands.) The design of each scene emerged through an iterative process that cycled between making observations about the data and an evolving set of basic compositional scene elements. To make sense of our stream of text data, we relied on Perl for text parsing and feature

extraction, some flavor of UNIX shell for process control, and the R programming environment for data analysis, modeling, and statistical graphics.

Why did you write your own software tools?

Given that the display "device" (the combined audio and visual components of the installation) was entirely new, we had little choice but to write our own software to control it.

For the most part, the software side of Listening Post *is constructed from what could be best described as "scripting languages." While it's a bit hard to pin down a precise definition for this term, it is often the case that such languages let you build up projects (programs or scripts) quickly in a fluid, interactive process that is distinct from programming in a "systems language" like C. For example, Perl is, by design, great for manipulating text (taking inspiration from previous UNIX shell facilities like awk); and over the years programmers and developers have created a stunning number of extensions to the language, including extensive tools for network programming. By working with Perl, I can output data to the VFD grid and quickly experiment with different scene dynamics, many of which involve parsing and computing with text. Using OSC, this same Perl program can also coordinate audio by sending messages to a Max/MSP process and to the TTS engine. Authoring scenes for* Listening Post *is an exercise in interprocess communication.*

Since 2000, the language Python has emerged as a strong competitor to Perl in this kind of application; Python even runs on many Nokia phones! If our development were taking place today, we would have to think seriously about programming in Python instead of Perl. The lesson here is that programming tools, and information technologies in general, are constantly in flux. If you choose software as a medium, your practice has to keep up with these changes. You need to be able to "read" a new language, assess its strengths and weaknesses, and determine which computations are "natural" (those that its designers have made easy to perform) and (if possible) why.

Why do you choose to work with software?

Software, or perhaps more generically computing, is the way I have come to know data.