The Sound of TouchNoise

Axel Berndt

University of Music Detmold

Center of Music and Film Informatics

Detmold, Germany

berndt@hfm-detmold.de

Nadia Al-Kassab and Raimund Dachselt

Technische Universität Dresden

Interactive Media Lab

Dresden, Germany

nadia.alkassab@gmail.com

dachselt@acm.org

Abstract

TouchNoise is a multitouch interface for creative work with noise. It allows the direct and indirect manipulation of sound particles which are added up in the panning and frequency space. Based on the mechanics of a multi-agent system and flocking algorithms, novel possibilities for the creation and modulation of noise and harmonic spectra are supported. TouchNoise underwent extensive revisions and extensions throughout a 3-year iterative development process. This article gives a comprehensive overview of the final TouchNoise concept, its mapping approach and interaction from

which a variety of unique sonic capabilities derives. Based on our experiences with a fully functional prototype implementation this article focusses on the systematic exploration and discussion of these novel sonic capabilities and corresponding playing techniques which differ strongly from traditional synthesis interfaces.

Multitouch, Agents, Noise

The advent of multitouch technology marks the beginning of a success story that made powerful computing technology ubiquitous. Devices can be found in many form factors from small watches, smartphones and tablets up to machine control panels, tables and walls. Due to the unified input and output space of an interactive surface, interaction with the contents on screen became more direct than through peripheral devices such as mouse and keyboard—users tap the visuals directly instead of performing input via off-screen devices and navigating pointers to the place of interaction. Touch displays offer a dynamic design space not only for custom graphical user interfaces and new interface widgets, but also for bimanual and gestural interaction techniques and multi-user settings that were formerly impossible or impractical. Along the lines of touching the data directly to exert an effect on it, interactive visualizations provide access beyond the traditional reliance on buttons, sliders etc. In the field of musical human-computer interaction, multitouch technology inspired many concepts and interface approaches that exploit new ways to access sound synthesis, sequencing and generative music systems.

With increasingly powerful computing technology a new type of digital musical instruments, the *active musical instrument*, emerged (Chapel 2003). Such instruments do not require the player to trigger each musical event (e.g. note) individually. They play autonomously in realtime. The player's role is to direct this process in a musically meaningful way. Among the numerous instances, we can name just a few examples

here. A typical representative is the Reactable interface (Geiger, Alber, Jordà, and Alonso 2010), a multitouch and tangible interface for sound synthesis and sequencing. Its technical framework has been adopted in the tabletop algorithmic composition system ReacTacT (Ó Nuanáin and O'Sullivan 2014). CollideFx is a multitouch patching environment by Gnegy (2014) for realtime sound synthesis and effects processing. Lopes et al.'s study with a multitouch DJing application attests an increase of interaction speed compared to a purely virtual (laptop) setup (Lopes, Ferreira, and Pereira 2011). The NodeBeat app (Sandler and Windl 2013) is a dynamic sequencer on the cusp of an interactive ambient music generator like several smartphone apps by Eno and Chilvers (2011, 2014).

Our system TouchNoise (Berndt, Al-Kassab, and Dachselt 2014) is a further example of a multitouch-based active musical instrument. It explores the range of perspectives in the creation of and interaction with stereophonic noise spectra based on an interactive particle system that is equipped with algorithms for different motion behaviors including Brownian motion, flocking, and flow fields. Multitouch gestures do not only set the parameters of these behaviors, but allow for directly influencing particles in various ways, such as exerting magnetic and repellent forces.

Multi-agent and particle systems have been the basis of several musical interfaces. Kuhara and Kobayashi (2011) present a kinetic particle synthesizer for mobile multitouch devices. Photophore is a synthesizer that applies a flocking algorithm to modulate up to 100 oscillators and create natural chorus effects (Dika 2014). SwarmSynth uses flocking to control a bank of oscillators in a 5D space (volume, pitch, pan, resonance, and noise), diversify their modulations, and create lively sounds (Hargreaves 2003). Orbits is a generative music interface that is based on an intuitive simulation of the movement of orbs and gravitational forces between them (Vera 2011). A deeper discussion of the use of interactive swarming in an improvisational music context is

delivered by Blackwell (2007). Artistic examples and C++ libraries for swarm-based music generation are provided by Bisig and colleagues in the Swarms project.[1] A more general discussion of the use of particle systems in audio applications—primarily addressing soundscapes (e.g. for games), DSP effects and granular synthesis—is given by Fonseca (2013). The SuperCollider Live Spatialization System (SCLiss) is a typical example that utilizes a particle system and certain motion characteristics such as Bownian motion, simple harmonic motion, and orbital motion (Pérez-López 2014). The author further provides a more dedicated state of the art review of spatialization systems. A further example of flocking-based sound spatialization is Kim-Boyle's (2008) spectral spatialization approach that utilizes the Boids algorithm by Reynolds (1987, 2007) and derives a (non-interactive) visualization similar to that in TouchNoise.

TouchNoise adopts such concepts in its multi-agent system of sound particles which is likewise based on the Boids algorithm, Brownian motion and further methods (see section "TouchNoise: An Overview"). Its conceptual core and most distinguishing aspect lies in the direct multitouch interaction with the particle system, the variety of mechanisms to influence the particles and the range of sonic capabilities that the mapping of input gestures to sonic reactions exploits thereby. This article contributes a systematic discussion and reflection on the novel and exciting prospects this approach allows with regard to sound aesthetics and sound interaction.

Throughout music history the role of noise as a shapeable musical material gained in importance. Drummers and percussionists developed a great mastery in the rhythmical use of differently colored noise spectra and established various sophisticated playing techniques. Noise became an important element in experimental music and musique concrète since the early 20th century. The era of electronic music introduced various noise-based effect sounds. The growing number of ever more flexible synthesis

[1] http://www.zhdk.ch/index.php?id=icst_swarms_e, last access: August 2016.

tools consolidated noise as an omnipresent sound material in electronic music.

Noise modulation is commonly based on stochastic signals synthesizing a somehow colored inharmonic frequency spectrum. Filters are then applied to further refine this spectrum. More complex frequency spectra, in fact any recorded sound, can be introduced via sampling and granular synthesis. All these techniques are well established and proved their practical worth over several decades of application. When looking at how they are used in music making and how the corresponding synthesis tools are handled, we recognize a prevalence of indirect interaction concepts. Sound manipulations are achieved by synthesis patch editing and parameter modifications of frequency, amplitude and filters plus certain distortion and waveshaping effects, all controlled via traditional control elements such as knobs, faders, and buttons. The mental effort of translating these controls to sounds in terms of predictive knowledge and usability is relatively high. This constitutes the motivation behind several research and development activities in musical human-computer interaction throughout the last decades in finding better mappings of control structures to sound synthesis (Hunt and Wanderley 2002) and reducing the dimensionality of the input space (Goudeseune 2002).

With TouchNoise we were striving for a more direct and immediate interaction to foster the creative and nuanced work with noise spectra and to provide a live performable instrument that opens up new perspectives for this purpose. Therefore, TouchNoise exploits the multitouch display as dynamic visual interaction domain and the particle system as basis for both, sound generation and dynamic visualization. User interaction becomes more direct as it takes place within this visualization of the sound, directly at the particles to be affected. The spatiality of the interaction domain parallels the spatiality of the particle system and its corresponding sound structures: the stereophonic frequency spectrum. As a close relative to this we see the “explorative synthesis interface” of the CataRT system (Schwarz, Beller, Verbrugghe, and Britton

2006). It places its sound material (grains for concatenative synthesis) as points in a two-dimensional descriptor space; triggering its playback is done by tapping into the space at or close to the sound material. Even though different in virtually every other aspect, TouchNoise and CataRT feature a similar kind of directness in accessing (or, in case of TouchNoise, affecting) their sound units by interacting in close proximity to their visual representation on screen.

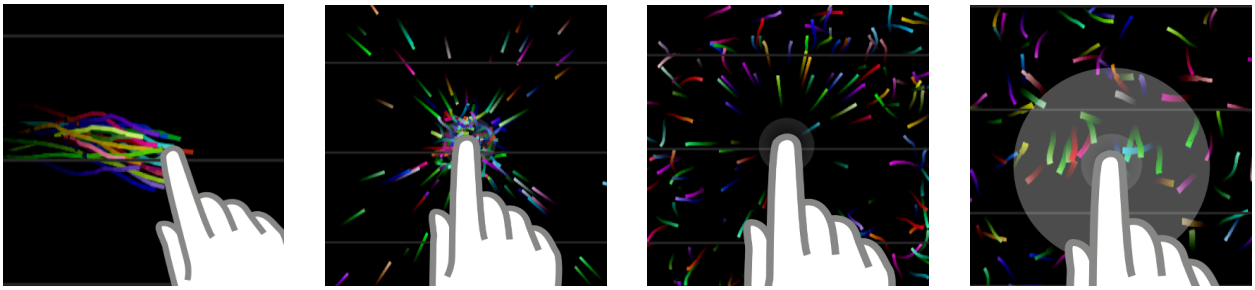
As a musical interface TouchNoise underwent a development process in which we pursued the following general goals, based on (McDermott, Gifford, Bouwer, and Wagy 2013). The instrument's basic concept should be easy to understand, supported by a direct correlation of auditory and visual output. Basic interactions should be very direct with no practical learning hump. Mastery of the instrument's full functionality exploits increasingly varied and complex sound patterns (open-endedness for long-term engagement, see Wallis et al., 2013) and demands the player to develop advanced, more nuanced gestures, and playing techniques (layered affordance). The visual impression should be well suited for live projection on-stage and roughly promote comprehension and virtuosity to the audience.

In the first, exploratory development phase we implemented the basic mapping and interaction concepts (Berndt, Al-Kassab, and Dachselt 2014). In practical tests and demo sessions (mainly with experts in human-computer interaction) we explored the sonic capabilities of the approach, gathered usability experiences, and collected user feedback for substantial revisions and extensions during the second development phase (Berndt, Al-Kassab, and Dachselt 2015). In section "TouchNoise: An Overview" we summarize the final concept, functionalities and technical implementation of TouchNoise. A comprehensive technical description is drawn by Al-Kassab and Berndt (2015). Section "The Sound of TouchNoise" then provides a systematic deduction and discussion of the various novel sonic properties that TouchNoise opens up through its specific mapping

and involvement of interactive multi-agent, flow field, and flocking mechanics. Such comprehensive analysis could not be given by previous, more technical publications. It marks the main contribution of this article and completes our series on the TouchNoise interface.

TouchNoise: An Overview

The basis of TouchNoise is a relatively simple visualization of the stereophonic frequency spectrum: The vertical axis represents frequency (up/down, high/low), the horizontal axis represents panning (left/right). A sine oscillator with a certain frequency and stereo position is visually represented as point on this plane and further referred to as *particle*. The plane is called the *particle playground*. When a particle moves along the horizontal axis, it changes its position in the stereo sound field. Movement along the vertical axis means the particle changing its frequency (between 20Hz at the bottom and 20kHz at the upper edge of the playground). Usually, multiple particles are present on the playground and add up to the stereophonic frequency spectrum. The particles are not static but by default perform Brownian motion (Nelson 2001) on the playground. The user/player can interactively add and remove particles and exert various influences on their motion, distribution and dynamics via multitouch gestures on the playground directly at the particles. This transforms the playground into an interactive visualization.



(a) Drag mode. (b) Magnetic mode. (c) Repel mode. (d) Accentuate mode.

Figure 1. Modes of direct touch interaction with the sound particles.

Different functions can be assigned to touch events, see figure 1. Thereby, it is possible to drag particles throughout the playground, attract and repel them to and from the touches, and accentuate their amplitude. Combinations of these functions are also possible, e.g. to attract particles and accentuate them when they come into a certain range. The radius of each effect can be set individually. Furthermore, it is possible to assign touch events with the creation or deletion of particles at and around the touch position. With the exception of accentuation, all interactions affect the particle distribution and (amplitude) mixing and take place directly within the stereophonic frequency spectrum or its visual representation on the touchscreen, respectively.

An upper and lower bar delimit the frequency axis of the particle playground and can be dragged to any position between 20Hz and 20kHz. The decision to achieve band-limiting in this way derives from experiments with the first prototype. The playground's vertical axis did initially range from 60Hz to 3200Hz. We soon realized that many interesting sound effects would go beyond this and successively increased the range. Even the borders of human perception can be interesting in a productive way as they allow artists to work with perceptual fading, i.e. letting particles disappear by leaving the perceivable range and fade in by coming back into it. The sound sample "Magnetic chords" on project page[2] gives a typical example.

Our main design goal with the TouchNoise interface was facilitating the direct interaction with this stereo noise spectrum. The TouchNoise interface is shown in figure 2. An accompanying demo video on the project page shows the interface at work. Nonetheless, there is still a number of indirect interactions in order to control the general characteristics of all particles. Two parameters influence the Brownian motion. The step width determines the particles' speed and the Brownian angle delimits the maximum rotation after each step. These two parameters set the dynamic behavior of the noise

[2]http://www.zemfi.de/research/touchnoise/

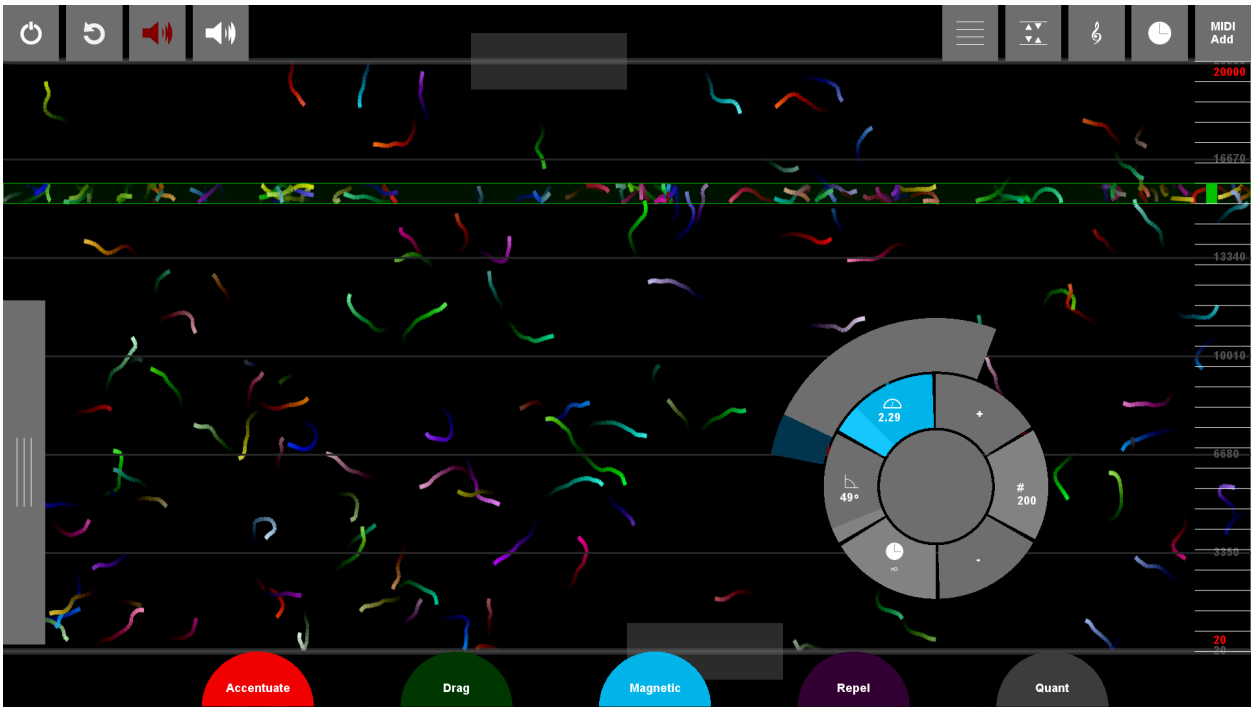
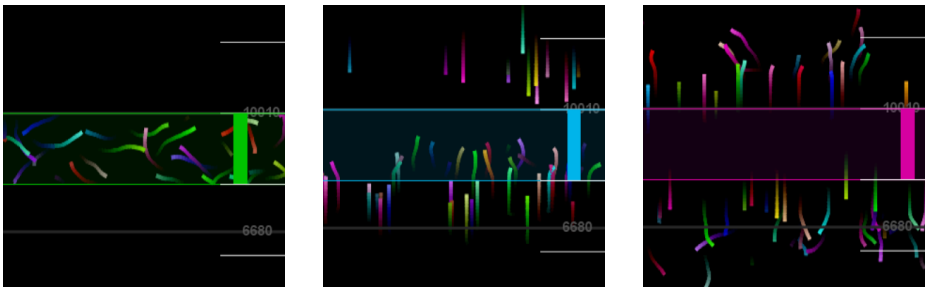


Figure 2. A screenshot of TouchNoise's multitouch user interface. A frequency band has an active drag function. Accentuation and magnetic mode are active for direct touch interaction within the particle playground. The radial particle menu is also opened; it can be dragged freely over the screen and is minimized when put at the border or flicked to it.

spectrum (e.g. static, slowly evolving, or brisk). A lifetime slider sets the timespan that each particle exists on the playground before being deleted automatically. A further slider allows the creation and deletion of particles at random positions.



(a) Frequency band drag mode.

(b) Magnetic frequency band.

(c) Repulsive frequency band.

Figure 3. Modes can be assigned to frequency bands.

The touch functions (drag, magnetic, repel, and accentuate, see figure 1) can, furthermore, be assigned to whole frequency bands (rows of the playground, see

figure 3) either through bimanual touch gestures on a "piano bar" at the right border of the screen or by using TouchNoise's MIDI connectivity. In the latter case, a MIDI keyboard can be used to trigger the frequency bands. When activated, a frequency band exerts the same effects as the corresponding touch function, i.e. it attracts or repels particles above and below within a certain radius, accentuates particles crossing the band, or holds them captive. The number of frequency bands and their size determine the discrimination of the frequency axis, from which different musical scales derive. Beyond the predefined scales, i.e. pentatonic, Pythagorean diatonic, and equal tempered chromatic, users can define their own scales. Once activated, the frequency bands may either remain active until the user deactivates them or they are automatically deactivated after an adjustable activation time. We introduced quantization as a further effect that can be assigned to frequency bands. Each particle that enters such a band plays its median frequency until it leaves the band again. In this way, tonality can be introduced into the sounds created.

The initial mapping of the vertical axis of the playground to the frequency domain is linear. Assuming a uniform distribution of the particles over the playground, the linear mapping causes an emphasis of higher pitches since the human subjective perception of pitch is approximately logarithmic in the frequency domain (Fastl and Zwicker 2007; Loy 2006). Thus, the noise sounds denser in higher pitches which is well suited for non-musical sound effects. TouchNoise allows for an even more nuanced definition of the frequency domain. The user can switch to a logarithmic scaling for a more musical use of TouchNoise and especially of the piano bar, as it allows a linear mapping of musical scales along the vertical axis. The particle motion, however, remains linear on the playground, i.e. uses a constant step width regardless of its vertical alignment. Hence, higher frequencies are traversed faster in the logarithmic mapping and the lower frequency bands are more emphasized. Further nonlinear distortions of the frequency domain are possible: the user can expand and constrict frequency bands,

see figure 4. While the particles still move visually with the same step width, the actual frequency steps vary according to the mapping of the visual position to the frequency position. The particles pass visually narrow frequency bands much faster (resulting in quick sweeps) than those being widened.

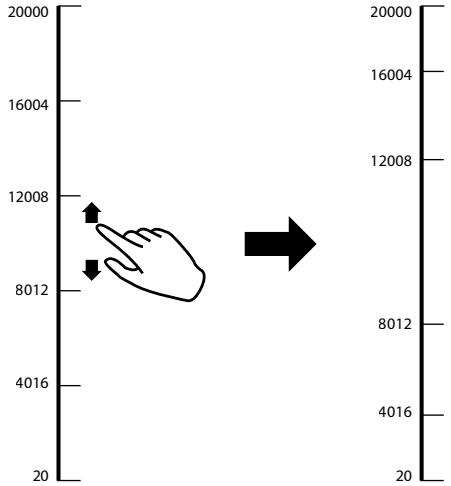
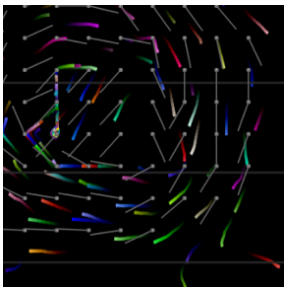


Figure 4. Frequency mapping distortion via pinch gesture.

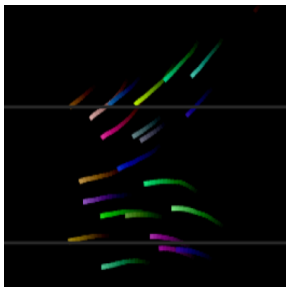
Up to now, the particle flow is defined by the parameters of the Brownian motion and influenced by magnetic, repellent, and drag interaction through touches or frequency bands. The particles switch back to Brownian motion when the interaction ends. The particle distribution fades gradually into white noise. This, however, does not suffice to define longer lasting directed flow. Therefore, we introduce flow field functionality to the playground. The flow field can be defined, literally painted, by drag gestures on the playground. Figure 5(a) shows an exemplary circular field. A particle that gets into it follows the indicated directions until it is released to the omnidirectional area where it switches back to Brownian motion.

We added different flocking functionalities to introduce persistence also to the clustering of particles. It is possible to add a leader particle to the playground that attracts particles nearby, see figure 5(d). Moreover, flocking and swarming are possible even without a leader through the Boids algorithm by Reynolds (1987, 2007). It gives

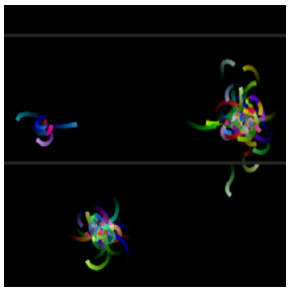
control over the three parameters separation (avoid crowding), alignment (head toward the average direction of local particles, see figure 5(b)), and cohesion (head toward the average position of local particles, see figure 5(c)).



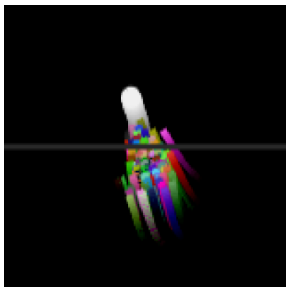
(a) A flow field can be used to direct the particle motion.



(b) Flocking by alignment.



(c) A strong cohesion causes the particles to stick together.



(d) The white leader particle attracts particles nearby.

Figure 5. Modes of particle flow.

All these influences—flow field, leadership, and Boids—can be weighted against each other and the Brownian motion. The influence of a flow field to the particle motion may, e.g. not completely eliminate Brownian behavior but, instead, the different step directions of each mechanism are weighted and interpolated. This allows for the creation of complex behaviors, e.g. a flock that follows a leader but the flock mates keep a certain minimal distance to each other and exhibit a certain degree of Brownian behavior even within the flock. The whole flock may follow a path through a flow field, but only roughly as the flow field has a low weighting. Touch interaction, however, has highest priority and dominates all other influences.

The technical implementation of TouchNoise has been done in Java. It utilizes the multitouch framework MT4j and the sound synthesis framework JSyn (Burk 1998). Each particle is synthesized by a sine oscillator. Frequency and amplitude modulations are smoothed by linear ramp envelope generators. Each particle further includes a level-difference stereo panning unit. Several hundred such particles can be instantiated and are added up to the system's output, see figure 6. Hardware performance limits the

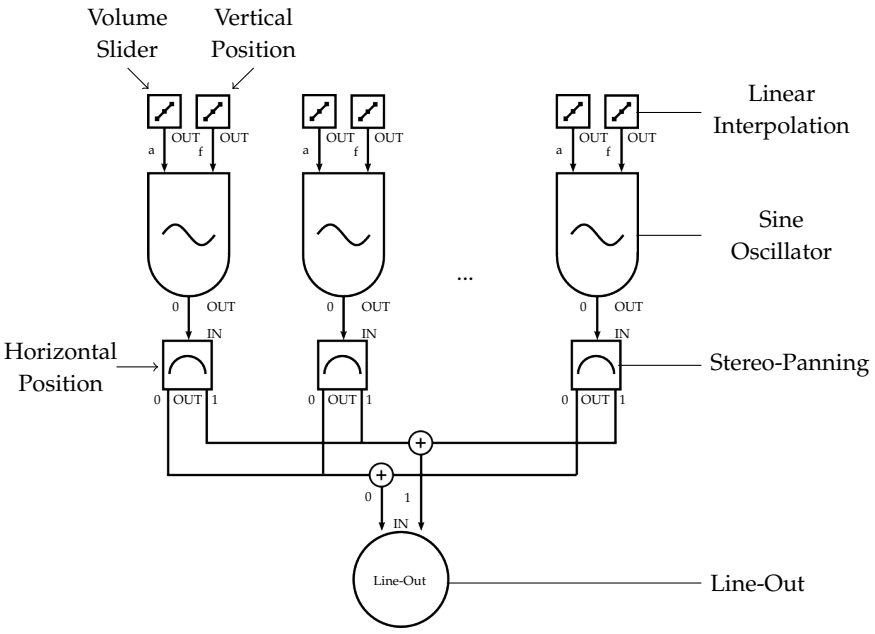


Figure 6. The sound synthesis of TouchNoise.

maximum number of particles: On a test system with Intel Core i7-3770 CPU, 16 GB RAM, GeForce GTX 680, and 3M multitouch screen we determined a limit of 200 particles for glitch-free performance. On faster systems we could go up to 400. Since previous publications focussed on the technical aspects and development documentation, the actual sonic properties of TouchNoise's mapping and interaction approach remained widely unreflected. It is the aim of this article, after summarizing the final state of TouchNoise, to draw a systematic derivation and discussion of its sonic capabilities and limitations throughout the next section.

The Sound of TouchNoise

For a systematic analysis and discussion of TouchNoise from a more aesthetic perspective, we followed methodologically a similar iterative approach that is used in interaction design. We tested systematically and experimented with single functions and aspects of TouchNoise first. Thereby, we assessed how accessible, easy and fast to use, and expressive these individual sound functions are. Where necessary, we had to

re-iterate and improve the interface or interaction with it. An example for this process has been described in the previous section for the frequency range and band-limiting of the particle playground. Later, we combined several functions to support more complex sound creation workflows and evaluated again how well their interplay worked. This whole process was dedicated to identifying the role that the functions, their combinations and TouchNoise as a whole can play in musical and non-musical contexts. In the following, we present the rich set of sonic capabilities of TouchNoise.

Particle Distribution and Dynamics

As long as only a few particles are spread over the playground, they are easily distinguishable as individual sine tones. This can be a basis for melodic use cases. The more particles are added to the playground—we experimented with up to 400 particles—and the higher their density throughout the whole playground is, the more diffuse is their sound, and it eventually fades into a homogeneous noise cluster.

An equal particle distribution throughout the whole playground resembles white noise. In traditional synthesizers this noise would be sent to one or more filters to shape its color. TouchNoise, however, has no such filters. Nonetheless, filter-like effects can be achieved by two different means: particle distribution and accentuation. Concentrating all particles in a tighter region resembles the sound of a very steep bandpass filter. Bandpass and bandreject filtering can further be achieved through accentuation (via direct touch and frequency band interaction) which sets clearly specified regions louder or quieter than the surroundings. Moving the accentuation area throughout the playground resembles the modulation of a traditional filter's cutoff frequency. Interestingly, a similar kind of effect can be achieved even without the amplitude as a differentiating parameter, but merely by density. Tight clusters with their denser sound contrast strongly with the surroundings, even though both have the same amplitude.

This shows that playing with the particle distribution on the playground marks the major creative means of TouchNoise—possibly even more prominent than the creation and deletion of particles or their amplitude-wise accentuation. The latter (accentuation), however, is a convenient means for complementary emphasis.

In this regard, the transition processes between different distributions mark a unique sonic property of TouchNoise which can hardly be reproduced with traditional synthesizer concepts. Typical examples are the dissipation of a tight cluster and the gradually increasing distinctness of the sound that a frequency band with a drag function produces with every particle it catches while the surrounding loses more and more sonic energy. These processes derive from the dynamics of the multi-agent system (Brownian motion, flocking, flow field). They may perform fast or slowly and can be influenced and designed in various ways by interaction, be it direct (drag, attract, repel via touch and frequency band interaction) or indirect (through the parameters of the Brownian motion and the weighting of the different flocking characteristics etc.).

In combination with the MIDI connection the effects can be combined in various ways, which allows the user to develop multifarious playing techniques. An example for this is what we call "Magnetic chords"[3]: The magnetic effect is assigned to the MIDI connection; playing chords on the keyboard triggers the corresponding frequency bands which then attract particles nearby. This creates a seamless fade from a noisy into a harmonic spectrum and back into noise after the key release. By creating and dragging particles, the user can redistribute sound energy between the chord notes played, assign a denser sound to some of them, and control the stereo direction of their sound. It may not even be intended that the particles reach a magnetic frequency band and aggregate into a salient cluster. Magnetic and repellent forces may be applied only to set particles in motion.

[3]See footnote 2.

Spatialization

TouchNoise, in contrast to other systems that are solely dedicated to sound spatialization (Kim-Boyle 2008; Pérez-López 2014), implements only level-difference stereo panning. Basically, the panning axis could be circularly extended to full surround. In order to achieve the same sonic density when the particles distribute over a wider space, a proportionate increase of particles is required, which rapidly becomes very demanding in terms of hardware performance. Furthermore, the current playground design does not represent the spacial circularity of a surround setup well—particles would jump from left to right edge and vice versa. In fact, the surround spatialization (as well as two- or three-dimensional setups of the sound field) are better served with a different mapping and interaction paradigm that requires less mental effort to translate input domain to output domain. Instead of multitouch gestures on a planar display we recommend spatial modalities such as free-hand gestures (Lech and Kostek 2013; Marshall, Malloch, and Wanderley 2009; Quinn, Dodds, and Knox 2009). TouchNoise is conceived and optimized for a stereo panning scenario. Its horizontal playground axis parallels the horizontal spatialization axis. From a music-aesthetic point we regard the frequency axis as the far more versatile and interesting playground dimension.

Frequency Mapping

Much effort has been invested in the mapping of the vertical playground axis into the frequency domain. This mapping has a major effect on the sound and tonality of TouchNoise. The linear mapping yields an emphasis of higher pitches and is well suited for effects sounds. The logarithmic mapping, by contrast, provides a more musical basis as it corresponds to the human perception of pitches and pitch intervals, respectively. This emphasizes the lower frequency range. Two sound examples on the project page[4] show typical use cases: “Funny lasers” is played on the linear mapping, “Pentatonic

[4]See footnote 3.

rain" on a logarithmic. Further nonlinear distortions by pinch gestures (see figure 4) may place the spectral emphasis on other frequency bands.

For understanding the resulting sonic properties, it is important to keep in mind that the particles move through the playground with a constant step width (as long as the user does not change this parameter). Except for the linear mapping, the speed at which the sine oscillators move through the stereophonic frequency spectrum (see section "TouchNoise: An Overview") is not constant. Compressed frequency bands on the playground are traversed with only a few steps which creates rapid sweep sounds. Expanded frequency bands on the playground require more steps to pass them, hence the sine oscillators perform slower glissandi.

With quantization we introduced a way to add tonality and create pitched sounds in TouchNoise. Quantized frequency bands are clearly audible in a non-quantized environment. The more frequency bands are quantized, the more the sound changes from noisy to tonal. If a single particle moves through a fully quantized playground, its output—a stepwise progression of scale tones—obtains melodic quality. Particles with limited lifetime and a predefined movement direction (using flow fields) may serve as motif generators. However, as this derives from the particle's motion, which is quasi-continuous, there cannot be melodic intervals other than to the neighboring scale tones. That means, it cannot jump to distant pitches without triggering those in-between. Classical monophonic lead/melody playing can be done via a MIDI keyboard that triggers magnetic frequency bands; now, the particles will always follow the MIDI notes. But they do not jump to the notes, they follow continuously which creates portamenti.

This shows that, even though TouchNoise can be played in a quasi-melodic fashion, it is no typical lead synthesizer and better suited for noise sounds, sound effects and tonal as well as noisy sound clusters and soundscapes. These can feature more or less complex inner dynamics—depending on the particle motion.

Persistent Sound Structures

The minimum step width that can be set for the particles is greater than 0. This means that the particles are always in motion. So are the sounds they produce. Due to the Brownian motion, any unequal distribution dissolves into an equal distribution after a certain period. This process may run faster with large step widths and narrow rotation scopes. It can be slowed down with only short step width and wide rotation scope ($\geq$ 180 degree) settings.

Beyond this, flow field functionality can be used to direct the particle motion without the need of explicit drag, magnetic or repellent interaction. With the various flocking properties (alignment, cohesion, separation, and following) more persistent particle clusters can be created which share common motion behaviors. Even though these mechanisms may in a way correspond to LFOs and envelopes (self-contained modulation sources) in traditional synthesizers, they create a more complex interplay between frequency and panning modulation in TouchNoise.

Frequency band interactions overrule flow field and flocking influences of the particle behavior, and direct touch interactions overrule frequency bands. Hence, it is always possible to intervene and change the defined characteristics.

What you can (not) do with TouchNoise

As the previous sections already show, the user of TouchNoise should not expect the behavior of a traditional musical instrument or synthesizer! TouchNoise follows its own unique approach of defining an interaction space filled with autonomous particles and corresponding affordances and possibilities to manipulate their behavior through multitouch gestures. In this it differs significantly from common note-wise instrument playing. Likewise, TouchNoise's distinct mapping of the particle system into a sounding output does not follow the conventions established by traditional synthesizers. In this

SwarmSynth (Hargreaves 2003), even though a keyboard-based note-wise approach, and CataRT (Schwarz, Beller, Verbrugghe, and Britton 2006), even though based on a different type of sound material and behavior, are conceptually the closest. Although TouchNoise's sound synthesis is based on additive synthesis with up to several hundred frequency modulated sine signals, the comparison with traditional additive synthesizers is problematic. Direct interaction with the particle system through multitouch gestures is the key part of the interaction concept, not keyboard playing even though the MIDI connection allows this, too. This provokes different music-aesthetic approaches. The following paragraphs expand on this.

TouchNoise is conceived as an active musical instrument, i.e. it creates its output autonomously. Instead of having to trigger each note or other type of musical primitive events separately, the user's role is to direct the instrument while it plays by itself. Even though some interactions can also be used for very immediate event generation (e.g. adding particles by touch gestures on the playground), this is not the usual pace of TouchNoise. The player triggers change processes and events that develop over time and feature a certain decay time (e.g. clusters that, once released, need time to spread out again). The user primarily interacts with the particles and their behavior. The creation and deletion of particles are mostly part of the initialization and termination of such processes.

Accordingly, TouchNoise's concept has been focussed on the work with sustained sounds. It is also possible to perform rhythmic patterns, e.g. with accentuation touches. Chimes-like percussion effects can be achieved by continuously creating particles with a lifetime of only 100 milliseconds. However, in general, TouchNoise is less suited for rhythmically accentuated, percussive playing due to its relatively flat envelope slopes at the creation, deletion, and accentuation of particles. The current implementation uses only attack-sustain-release envelopes. Experiments with other characteristics are due.

We expect that attack-decay shapes could be especially relevant in combination with limited particle lifetime: Particles would get a clear accent at their creation and continuously fade out until deletion. Furthermore, accentuation can be enhanced by a more percussive peak at the start of the amplitude increase.[5]

When a MIDI keyboard is used to accentuate or attract particles within and around certain frequency bands, a further peculiarity of TouchNoise emerges. As a consequence of the particle distribution, not every pitch is always playable. Accentuating a region that has no particles has no audible effect. Attracting particles towards the lowest and highest frequency band of a chord causes the bands in-between to stay empty. Again, interaction with the particle behavior and distribution is central to the TouchNoise concept. The musician not only triggers the instrument but has to interact with its specific behavior. Playing magnetic frequency bands on a MIDI keyboard may not even intend to play specific pitches but to affect the surroundings, to attract particles into a certain direction.

Furthermore, each particle represents a sine oscillator that outputs sound energy at its specific frequency and stereo position with no overtones. Particles are never coupled in terms of a fixed overtone structure, so that they could establish a consistent timbre. Also piano playing with the frequency bands affects only these frequency bands and creates no harmonics, i.e. higher frequency content. Against this background, the concept of timbre as a spectrum of overtones over a fundamental frequency does not apply to TouchNoise. Instead, another musically expressive design axis derives from the particle distribution: noise clusters can be condensed until they transform into pitched sounds and vice versa.

Among the strengths of TouchNoise are its unique possibilities to interact with

[5]This is, of course, not true for accentuation levels below the basic volume level, i.e. when the accentuation attenuates the output of affected particles.

noise spectra, to create sound effects such as sweep sounds and to modulate and transform the sounds in ways that are hardly possible with traditional synthesizer concepts. TouchNoise can also be used for tonal music, e.g. to create slowly evolving pad sounds, tonal/pitched sounds surrounded by noise or that fade into noise when released (as in the above mentioned magnetic chords example).

TouchNoise's output does not need to be the end of the DSP chain. As with many musical instruments in a greater music production context, it is convenient to refine the sounds created. Equalizing can attenuate TouchNoise's harsh treble. Reverb effects can introduce a certain spatiality to the initially dry sound. TouchNoise might also be used to deliver the basic sound material for a traditional subtractive synthesis. Furthermore, in many situations less than the maximum number of particles is audible, only a few particles may have been created. In these situations the loudness level is significantly lower than a full tutti of several hundred particles. If this is not desired, a subsequent dynamic range compression helps compensating this.

Conclusions

It is important to understand TouchNoise as a self-contained musical instrument with its own unique properties and possibilities. TouchNoise does not resemble another instrument's conception. Just as it is practically not feasible mimicking another instrument with TouchNoise, TouchNoise's sound, behavior and capabilities cannot be mimicked by other instruments.

TouchNoise is based on the additive synthesis of up to several hundred sine signals represented by particles, autonomous individuals in a multi-agent system, that spread throughout the stereophonic frequency spectrum. The user interacts with the particles by means of rich and intuitive multitouch interaction and the MIDI connection and thereby affects the sounds they produce. This article gave an introduction to TouchNoise's

concept and functionalities. We provided a comprehensive, systematic analysis and discussion of the novel sonic capabilities that derive from this technical basis and interaction approach. We described several examples for typical playing techniques.

TouchNoise's strength lies in its openness that allows the creation of complex effects and playing techniques through various combinations of touch gestures, frequency bands and MIDI connection, flocking and flow field mechanics. These open up a variety of possibilities for creative musical-aesthetic experiments and new sounds.

References

Ó Nuanáin, C., and L. O'Sullivan. 2014. "Real-time Algorithmic Composition with a Tabletop Musical Interface—A First Prototype and Performance." In *Audio Mostly 2014: 9th Conf. on Interaction with Sound—Imagining Sound and Music*. Aalborg University, Interactive Institute/Sonic Studio Piteå, Aalborg, Denmark: ACM.

Al-Kassab, N., and A. Berndt. 2015. "TouchNoise: A Multitouch Noise Instrument." In A. Berndt, (editor) *Works in Audio and Music Technology*, chapter 2. Dresden, Germany: TUDpress, pp. 31–57.

Berndt, A., N. Al-Kassab, and R. Dachselt. 2014. "TouchNoise: A Particle-based Multitouch Noise Modulation Interface." In *Proc. of New Interfaces for Musical Expression (NIME) 2014*. London, UK: Goldsmiths, University of London, pp. 323–326.

Berndt, A., N. Al-Kassab, and R. Dachselt. 2015. "TouchNoise: A New Multitouch Interface for Creative Work with Noise." In *Audio Mostly 2015: 10th Conf. on Interaction with Sound—Sound, Semantics and Social Interaction*. Aristotle University of Thessaloniki, Thessaloniki, Greece: ACM.

Blackwell, T. 2007. "Swarming and Music." In E. R. Miranda, and J. A. Biles, (editors) *Evolutionary Computer Music*, chapter 9. London, UK: Springer, pp. 194–217.

Burk, P. 1998. "JSyn – A Real-time Synthesis API for Java." In *Proc. of the Int. Computer Music Conf. (ICMC)*. Columbus, Ohio, USA: International Computer Music Association, Ohio State University.

Chapel, R. H. 2003. "Realtime Algorithmic Music Systems From Fractals and Chaotic Functions: Towards an Active Musical Instrument." Master's thesis, University Pompeu Fabra, Department of Technology, Barcelona, Spain.

Dika, N. 2014. "Photophore Synth." Taika Systems Ltd. App on iTunes, v1.0.

Eno, B., and P. Chilvers. 2011. "Bloom." Opal Limited. App on iTunes, v2.1.1.

Eno, B., and P. Chilvers. 2014. "Scape." Opal Limited. App on iTunes, v1.1.

Fastl, H., and E. Zwicker. 2007. *Psychoacoustics: Facts and Models*, *Information Sciences*, volume 22. Berlin, Heidelberg, Germany: Springer, 3rd edition.

Fonseca, N. 2013. "3D Particle Systems for Audio Applications." In *Proc. of the 16th Int. Conf. on Digital Audio Effects (DAFx-13)*. Maynooth, Ireland: National University of Ireland.

Geiger, G., N. Alber, S. Jordà, and M. Alonso. 2010. "The Reactable: A Collaborative Musical Instrument for Playing and Understanding Music." *Her&Mus. Heritage & Museography* :36–43.

Gnegy, C. N. 2014. "CollideFx: A Physics-Based Audio Effects Processor." In *Proc. of New Interfaces for Musical Expression (NIME) 2014*. London, UK: Goldsmiths, University of London, pp. 427–430.

Goudeseune, C. 2002. "Interpolated mappings for musical instruments." *Organised Sound* 7(2):85–96.

Hargreaves, L. 2003. "SwarmSynth v1.01." AnarchySoundSoftware. VST plugin.

Hunt, A., and M. M. Wanderley. 2002. "Mapping Performer Parameters to Synthesis Engines." *Organised Sound* 7(2):97–108.

Kim-Boyle, D. 2008. "Spectral Spatialization—An Overview." In *Proc. of the Int. Computer Music Conf. (ICMC)*. Belfast, Northern Ireland: International Computer Music Association, Sonic Arts Research Centre, Queen's University Belfast.

Kuhara, Y., and D. Kobayashi. 2011. "Kinetic Particles Synthesizer Using Multi-Touch Screen Interface of Mobile Devices." In *Proc. of New Interfaces for Musical Expression (NIME) 2011*. Oslo, Norway: University of Oslo, Norwegian Academy of Music, pp. 136–137.

Lech, M., and B. Kostek. 2013. "Testing A Novel Gesture-Based Mixing Interface." *Journal of the Audio Engineering Society* 61(5):301–313.

Lopes, P., A. Ferreira, and J. A. M. Pereira. 2011. "Battle of the DJs: an HCI perspective of Traditional, Virtual, Hybrid and Multitouch DJing." In *Proc. of New Interfaces for Musical Expression (NIME) 2011*. Oslo, Norway: University of Oslo, Norwegian Academy of Music, pp. 367–372.

Loy, G. 2006. *Musimathics: The Mathematical Foundations of Music*, volume 1. Cambridge, Massachusets: MIT Press.

Marshall, M. T., J. Malloch, and M. M. Wanderley. 2009. "Gesture Control of Sound Spatialization for Live Musical Performance." In M. Sales Dias, S. Gibet, M. M. Wanderley, and R. Bastos, (editors) *Gesture-Based Human-Computer Interaction and Simulation*, *Lecture Notes in Computer Science*, volume 5085. Berlin, Heidelberg, Germany: Springer Verlag, pp. 227–238.

McDermott, J., T. Gifford, A. Bouwer, and M. Wagy. 2013. "Should Music Interaction Be Easy?" In S. Holland, K. Wilkie, P. Mulholland, and A. Seago, (editors) *Music and Human-Computer Interaction*, chapter 2. London, UK: Springer, pp. 29–47.

Nelson, E. 2001. *Dynamical Theories of Brownian Motion*. Princeton, NJ, USA: Princeton University Press, 2nd edition.

Pérez-López, A. 2014. "Real-Time 3D Audio Spatialization Tools for Interactive Performance." Master's thesis, Universitat Pompeu Fabra, Barcelona, Spain.

Quinn, P., C. Dodds, and D. Knox. 2009. "Use of Novel Controllers in Surround Sound Production." In *Audio Mostly 2009: 4th Conf. on Interaction with Sound—Sound and Emotion*. Glasgow, Scotland: Glasgow Caledonian University, Interactive Institute/Sonic Studio Piteå, pp. 45–47.

Reynolds, C. W. 1987. "Flocks, Herds and Schools: A Distributed Behavioral Model." *ACM SIGGRAPH Computer Graphics* 21(4):25–34.

Reynolds, C. W. 2007. "Boids: Background and Update." URL http://www.red3d.com/cwr/boids/. Accessed: 20 Jan. 2015.

Sandler, S., and J. Windl. 2013. "NodeBeat." app for iOS, Blackberry, Android, and Windows.

Schwarz, D., G. Beller, B. Verbrugghe, and S. Britton. 2006. "Real-Time Corpus-Based Concatenative Synthesis with CataRT." In *Proc. of the 9th Int. Conf. on Digital Audio Effects (DAFx-06)*. Montéal, Canada, pp. 279–282.

Vera, B. 2011. "Orbits—Generative Music." Android app.

Wallis, I., T. Ingalls, E. Campana, and C. Vuong. 2013. "Amateur Musicians, Long-Term Engagement, and HCI." In S. Holland, K. Wilkie, P. Mulholland, and A. Seago, (editors) *Music and Human-Computer Interaction*, chapter 3. London, UK: Springer, pp. 49–66.