Creative Sound Modeling with Signed Distance Fields

Tendsin Mende
Lars Engeln
Matthew McGinity
Rainer Groh
tendsin.mende@tu-dresden.de
lars.engeln@tu-dresden.de
matthew.mcginity@tu-dresden.de
rainer.groh@tu-dresden.de
Technische Universität Dresden, Institute of Software and Multimedia Technology
Dresden, Germany

ABSTRACT

Wavetable synthesisers often use 2D or 3D surface representations to visualise the internal table of values that produce the final audio waveform. This representation can be used to extend the expressiveness of synthesizers, by allowing more intuitive creation and manipulation of the 3D forms. In this work we explore the use of Signed Distance Fields (SDF) for representing and visualising 3D wavetables. SDF representations allow intuitive interactive sculpting and animation of 3D forms. We demonstrate different methods for transforming these forms into sound in real-time, by intersecting arbitrary 2D surfaces with the 3D forms and interpreting the results as sound, in both time and frequency domains. This extends the concept of wavetable synthesizers, and as SDFs lend themselves to real-time, immersive display and interaction through motion and gesture, the technique can be used to construct novel interfaces for musical expression.

KEYWORDS

Wavetable, Synthesizer, SDF, Signed Distance Field, Auralisation

1 INTRODUCTION

While digital sound synthesis has certainly expanded the palette of sounds available to the digital composer, the discovery, construction or manipulation of novel and complex sounds is often limited by the interface provided for controlling the underlying synthesis system [11]. Often digital synthesizers allow manipulation of only a limited number of parameters, restricting the searchable space of possible sounds and limiting creative expression [11, 17, 18]. Nonetheless, there are approaches for common synthesizer types to emphasize the creativity, for instance, with geometric Phasors [13], touch controlled waveshaping [16] and granular synthesis [20], or with advanced techniques for spectral editing [1], as well as allowing real-time control inside Virtual Reality [2]. Such systems explore extensions of known concepts by providing more intuitive, natural interfaces for controlling systems with many degrees of freedom.

MuC'22, 04.-07. September 2022, Darmstadt

https://doi.org/10.18420/muc2022-mci-ws03-339

Here, we introduce a wavetable synthesizer based on the scanned synthesis approach (cf. [8]) using Signed Distance Fields (SDF) to represent 3D form and structure. The 3D structures are translated into sound using a variety of scan-line slicing techniques [5]. Some contemporary synthesizers provide a 3D visualisation of their internal wavetable. Some allow for modification on a column-by-column basis, but such an approach to sound modelling is limited by ease and speed with which complex wavetables can be constructed. It takes a lot of time to create desired interesting effects that subsequently distract from the task of making music or sounds. Previous work has demonstrated the use of simplex noise to fashion 2D terrain-like wavetables [7]. Methods for spectral-editing of sounds in the frequency domain using tools commonly found in image processing demonstrate another approach to graphical sound modelling [1, 2].

2 SIGNED DISTANCE FIELDS FOR SOUND MODELING

When considering sound synthesis as a problem of 3D shape representation and manipulation, a wide variety of and techniques commonly found in 3D graphics become applicable. While polygons or voxels are perhaps the most well established representations, Signed distance functions [2] provide a number of important advantages. Firstly, they allow for constructive modelling such as additive, subtractive, intersection and modulo operations that are relatively difficult to achieve with polygonal representations. Secondly, they can easily represent smooth and soft intersections and conjunctions between shapes - a form of "virtual clay", and a small set of simple primitives and operators can be combined intuitively to create arbitrarily complex shapes. Recently, SDF modeling has been demonstrated to great effect in popular game [10] and modeling software [3]. Despite the limited number of atomic shapes and operators (i.e. add, sub, smooth), users of these platforms have demonstrated an extraordinary level and range of creative expression. This provides support for the notion that a small palette of elements that can be combined to create ever more complex forms may offer a more intuitive approach to 3D modelling than other representations, and allow an artist to develop their own visual language (compare with [9]). Furthermore, as SDFs lend themselves to efficient rendering on a GPU, they allow for real-time manipulation.

The 3D SDF is rendered sonically by projecting a line onto the 3D SDF, which can be conceived as taking a 2D slice of the SDF.

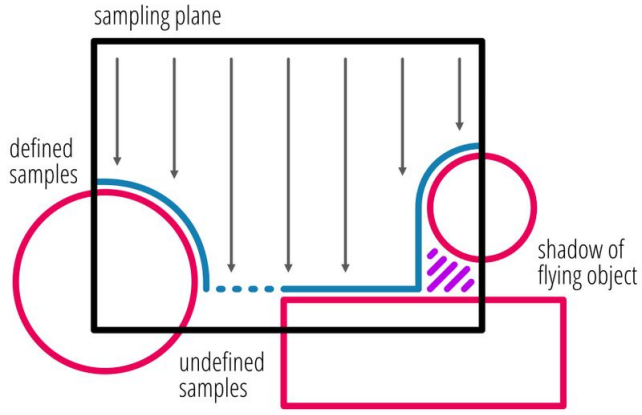

Figure 1: Sampling a 3D Object for wavetable synthesis top-down on a plane. Emphasizing undefined samples and flying objects creating *shadow*.

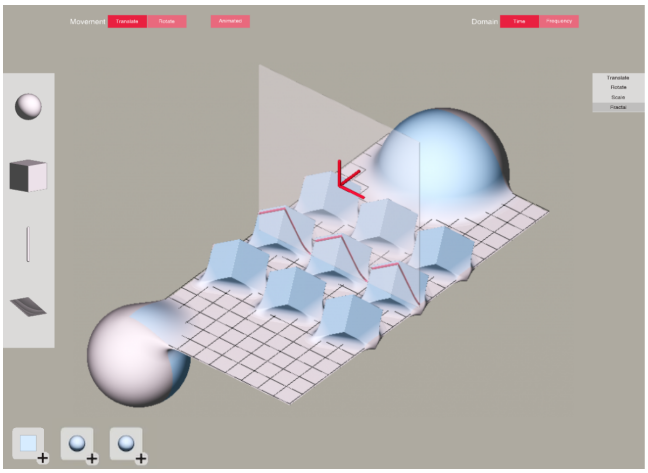

Figure 2: Primitives for dragging in (left), history (bottom), settings (top), widget for sharping/smoothing a primitive (center), influencing the sampled area with handles (red).

This yields a 1D array of "height values", in which the horizontal dimension is mapped to either time or frequency (c.f. Fig. 1). These height values are read directly as amplitudes in the time domain, or alternatively as amplitudes in the frequency domain, representing the spectral composition of the sound. The start and end of this line are considered as begin and end of one period in the respective space. In the time domain this means 2π, in frequency space this can lead to a range of $0Hz - 20.000Hz$ (or similar). In practice the samples have to be post processed to create an intuitive translation. The time domain needs to smooth out the transition between the 2π boundaries to prevent audible *clicking* noises. In the frequency domain samples need to be maped to a human auditory friendly scale, like the Mel-Scale [19] [15] since human perception does not scale lineraly but based on dissonance and overtone similarities in this space [12].

The power of our system (Fig. 2) is seen by contrasting with common 2D wavetable synthesizers, which usually sample only one row in their wavetable at a time. As such, they can only "slice" the wavetable orthogonally. In our system, however, the 2D slice may be taken in any 3D direction or position. Similarily, while a traditional 2D wavetable synthesizer is often constrained to move the slice orthogonally along either the rows or cols of the table, in our system the 2D slice can be animated with arbitrary rotations and translation.

Depending on the model multiple edge cases can arise (compare 1). Undefined samples can be created by cutting holes into models. Flying objects create un-sampled *shadowed*regions. Both cases can be handled in a intuitive way depending on the audio domain. Undefined samples are treated as zero-height samples in both domains. Sampling flying objects result in *jumps* in the sample buffer. Depending on the intended result those jumps can be smoothed.

By simply sampling in one direction, flying objects yield interesting possibilities when put in motion. When using SDF smooth blending operators, they can smoothly emerge and separate from other objects, creating a transition from hard borders.The shaded areas also offer potential as the sampling plane rotates, exposing structures over time.

In the time domain, where the sampled height values can be sent directly to the audio device for audification, there are two important properties: the sampling rate and the currently played key of any musical controllers. Depending on the key, the resulting sound shall have a corresponding higher or lower sound. By former definition the extent of the sampling plane is 2π wide. Therefore, when playing the *A4* (~440Hz) the plane sampler walks over the sampled curve $440hz/2\pi =\sim 70$ times per second. When not playing a key based instrument the sampling speed (formerly set by the played key's frequency) is set by the synthesizer. In addition, detecting the current sampled pitch and shifting it (i.e. with a phase vocoder) to the desired key is possible.

In practice, just taking the values of the sampler can create discontinuities when concatenating sets of derived samples. Those discontinuities can be removed by fading begin and end of the sample of each window to guarantee a smooth transition. The strength of overlapping can be controlled interactively. For fading, either linear blending of the last n samples of the last buffer and the first n samples of the following buffer, or any weighting function like a sigmoid or known window functions can be used. Samples on the upper boundary of the sampling plane are treated with soft-clipping.

In the frequency domain, the sampled height values are interpreted as frequency intensities, and so to play the sound, a transformation into the time domain must first take place. This is done via an inverse discrete Fourier transformation (iDFT). In terms of the resulting pitch and desired key, the resulting time domain samples are handled as described above with time domain samples. Alternatively, the sampled spectrum is shifted prior to the iDFT, such that the highest intensity matches the desired frequency, with assumption that the loudest frequency is perceived as the pitch.

A common way to animate the slicing of wavetable syntheziser (which is mentioned here as linear sampling) is, to animate/move over time. Patterns are possible, like sweeping back and forth, or just repetitively in one direction. The animation speed is responsible for

creating new characteristics in the final sound. Since it is not bound to just translation, rotation in any direction or scaling of the *cutting plane* can also be animated over time (see Fig. 3). Another common source for animating values is key pressure of input instruments, often encoded as the velocity of key presses in the MIDI standard. Another interesting source is air pressure in wind-MIDI-controllers. Based on those, effects like Vibrato can be achieved and modeled. Theoretically any function that gathers an amount of data from a 3D model would be possible.

2.1 Editing the SDF

A drag and drop interface allows geometric primitives to be selected and added to the scene (compare Fig. 2, left). The panel on the left displays a list of the primitives in their unmodified form, without scaling, coloring or transformation. Color represents the *impact* of a primitive, meaning either loudness in time domain, or frequency intensity in the frequency domain.

The *impact* translates directly to the height of each primitive relative to the current sampling technique. Smoothing the primitive's edges or molding together with other primitives at runtime is achieved with a widget (see Fig. 2, center). Every primitive is put into the edit history stack (see Fig. 2, bottom).

The two main parameters of the automation are the response of the sampling plane to time or any MIDI input, and the way the sampling layer moves. There are two motion types, moving/translation and rotation of the sampling plane. Those are toggled in the control bar (see Fig. 2, top).

Time based movement allows to move the plane continuously based on time. For simplicity, this only includes movement on the normal axis of the plane (similar to traditional wavetable synthesizers), as well as rotation around the planes center on the upwards pointing axis of the scene.

The primitive selection is the starting point for most use cases. The current implementation includes the four main primitives: sphere, box, plane and line. The plane is reduced from an infinite plane to a square (bound infinite plane). The implementation permits readily the introduction of additional primitive types in the future. A simple drag and drop allows addition of new primitives into the scene, and existing primitives can be selected and modified at will.

The type of primitive can, as well as the type of combination with the scene (union or subtraction) be changed in the edit history. The history displays the order in which the primitives are applied to the scene as well as its combination type (adding/union, subtracting) with the former primitives. Each primitive can be changed by selecting its presentation in the edit history and cycling through all available primitives. By clicking the combination type icon in the lower right, the type is switched between union and subtraction. Furthermore, the order of edits can be changed by dragging the primitives panel within the edit history. Other primitive properties like smoothness, translation and rotation are set within the 3D view. Roundness of the object as well as smoothness of the combination with the scene are changed within the 3D view via the *smoothness control* as a draggable widget, which placed as overlay next to the current selected primitive.

With a fractal like splitting tool an object is divided to a 3x3 instantiation (compare Fig. 3), which will scale and rotate likewise with the original object (compare rotated cubes in Fig. 2). The distances between the instances are set with the splitting tool itself. Thereby, a rapid way to create and explore structure as soundscapes are achieved. Currently, the sampling line is marked in the 3D world via a line strip based on the sampling state. For indicating the sampling speed by visualizing the current sampling point, might provide better feedback especially when considering a circular shaped sampling. Both domains are implemented. The time domain implementation is located directly in the synthesizer implementation. The frequency domain is added into its own library that handles caching and multi-threading. At runtime, the time domain is split into two main steps. The first one generates a buffer of sampling points based on the current sampling state. The second step uses those sampling points to trace rays through the model. This procedure is executed for each voice. The resulting intersection locations are filtered for missing values and normalized into the range $-1.0..1.0$ to fit the VST protocol. Afterwards the sample buffer of each voice is summed up and normalized via the sigmoid function. Missing values are currently always substituted with a sample value of 0.0. An additional per-voice search and a replace filter would be needed. Fading of previous and new audio buffers is currently not needed. The only per voice state that is tracked is the phase of each voice. At runtime a step size based on the voices frequency and sampling rate of the synthesizer is calculated. The step size is used to calculate the ray origins on the sampling ray.An orthogonal camera is used to keep lengths and angles in the whole screenspace to guarantee comparability of objects in the whole scene. The camera has a orbital control and can only rotate within the upper half-sphere of the scene by bounding the cameras pitch between $0°..90°$.

3 DISCUSSION

At the moment *sphere tracing* [6] is used for sampling the implicit surface, but also *segment tracing* [4] for even higher efficiency is considered. Our synthesizer is able to switch the audio domain from sampling the time domain to frequency domain on a per-audio-frame basis. Before filling a new audio frame the inner state is updated based on messages received from the interface. Depending on the inner domain state and currently active voices, the requested audio buffer is filled.

The resulting signal after sampling is an exact representation of the sampled surface within the synthesizer (note that the resulting signal is heavily modified by the sigmoid function). An unforeseen characteristic of the signal in practice is that it often sounds like a rectangular or sawthooth wave. The characteristic can be explained based on two observations. The first being the sudden sample value jumps when having non smoothed or flying objects in the scene. The second being the sigmoid's property to distort higher sample values more than lower ones. The distortion effect can be reduced by lowering the primitive (or moving the sampling plane upwards). This reveals the original model better in the final signal, but reduces the volume range of the signal. On the other hand, this gives more expressiveness by accident, being able to achieve sine like sounds,

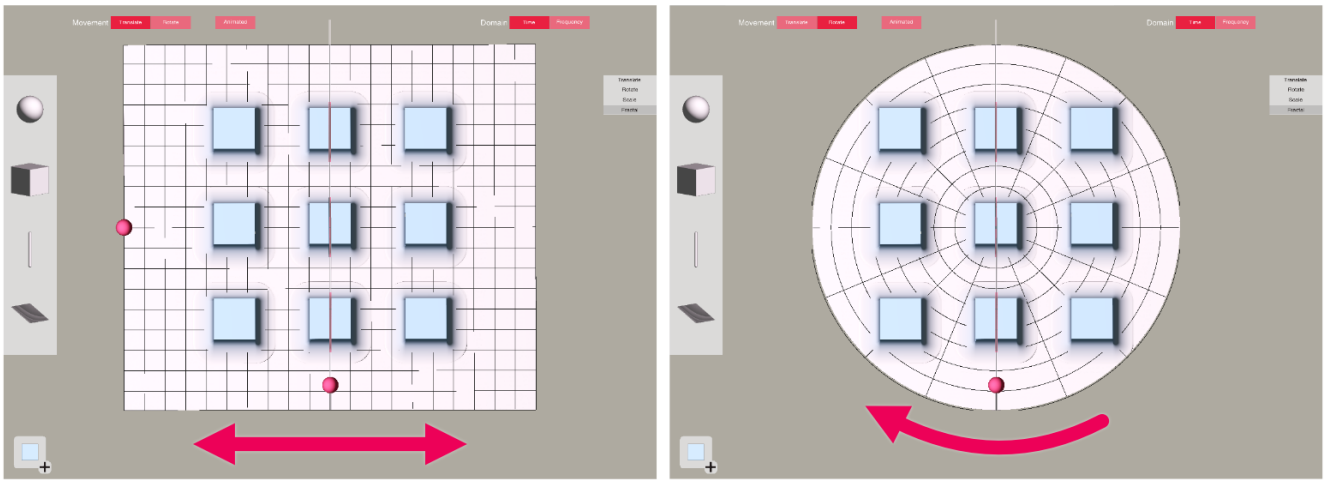

Figure 3: The view from top unveils the manipulation of the sampling plane's width and the used area for automated movement. The automation via transition (left) and via rotation (right) is supported by giving a grid for the corresponding area.

noisy soundscape, up to clear but overtone rich sawthooth like wave; in addition to a overdrive if moved to the top.

In frequency domains, multiple similar frequencies can be active at a time. This can create a sound like having a reverberation with a small room size. This effect is difficult to control which is currently the biggest disadvantage of this domain. A better intensity scaling as proposed by the concept might improve this. The unintended sound effect can be reduced by reducing the sampling resolution, which in turn creates bigger space between activated frequencies. The disadvantage is that frequency activations are audible when moving either primitives or the sampling plane.

While testing the prototype, first patterns of primitive combinations emerge. In the time domain the most notable patterns spheres that are smoothly added to the scene. This usually results in a soft sine like sound. Cubes create a sudden sample value changes, which results in the already mentioned rectangular wave sound. The effect can be weakened by increasing the *smoothness* factor. In the frequency domain small-scaled primitives proof to be more controllable compared to their larger counterparts. An interesting pattern are *V-like* shapes on the horizontal plane when combined with a moving sampling plane. The shape becomes audible through two main frequencies that either split up, or come together over time depending on the animation direction, which intuitively is the equivalent of a visual *V* as a sound.

It would be desirable if these were available as saved and thus reusable templates. In the future, other types of primitives and sound objects should be explored. Smooth primitives like *Non-uniform rational B-spline* (NURBS) [14] surfaces might allow better modeling sinus like waves in the time domain.

The prototype is currently only tested by people somehow involved in the project with varying experience in classic digital sound production. The reception of a wider audience remains to be seen. At the current state the ease of exploring new sounds can already be observed. Compared to classic wavetable interfaces the exploration process could be understood less as a refinement process of a given preset and more as *trial and error*. For a live setting motion controls could be interesting, those are not implemented in the current prototype however.

The SDF-based system presented here offers a range of use-cases, from the common desktop use with mouse and MIDI-keyboard in a home or professional studio setup, to the use of common hand tracker in that studio context, to the use of head mounted displays (HMD) for XR (eXtended Reality) with integrated hand tracking.Working in XR might be still seated at or near the desktop for intensely using a DAW or take place in a larger space for exploring sound. In addition, live modeling of SDFs with large physical gestures and movements lends itself to live performance. For example, a choreography or improvisation might explore the interdependence of embodiment, movement and sound. The system also allows for real-time collaboration between musicians with a division of roles, such as one performing sound modulation (sound design) with another attending to sound composition (musician).

4 CONCLUSION

This work presents an extension of the wavetable synthesizer. Synthesis in both the time and frequency domains are identified as viable. The 3D representation of wavetables is used as a basis for extensions to create a more intuitive user interface. Based on SDFs a 3D modeling technique for the creation of 3D surfaces is conceived. The technique borrows aspects from known 3D modeling software and improves them for wavetable modeling.

Ways of interpreting the modeled surface are explored with automation areas in space. Thereby, the known sampling strategy of wavetable synthesizers are extended by discussing the *linear sampling*. Known automation methods are extended for the gained degrees of freedom as well. Both, the interpretation methods and their automation are not explored to their full extent. Further extensions and new methods could be explored similarly.

REFERENCES

[1] Lars Engeln and Rainer Groh. 2019. CoHEARence: A Qualitive User-(Pre-)Test on Resynthesized Shapes for Coherent Visual Sound Design. In *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound* (Nottingham, United Kingdom) *(AM'19)*. Association for Computing Machinery, New York, NY, USA, 98–102. https://doi.org/10.1145/3356590.3356606

[2] Lars Engeln, Natalie Hube, and Rainer Groh. 2018. Immersive VisualAudioDesign: Spectral Editing in VR. In *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion* (Wrexham, United Kingdom) *(AM'18)*. Association for Computing Machinery, New York, NY, USA, Article 38, 4 pages. https://doi.org/10.1145/3243274.3243279

[3] ephtracy. 2021. *MagicaCSG*. https://ephtracy.github.io/index.html?page=magicacsg

[4] Eric Galin, Eric Guérin, Axel Paris, and Adrien Peytavie. 2020. Segment Tracing Using Local Lipschitz Bounds. *Computer Graphics Forum* (2020). https://doi.org/10.1111/cgf.13951

[5] Shawn Greenlee. 2013. Graphic Waveshaping. In *13th International Conference on New Interfaces for Musical Expression, NIME 2013, Daejeon, Republic of Korea, May 27-30, 2013*, Woon Seung Yeo, Kyogu Lee, Alexander Sigman, Haru (Hyunkyung) Ji, and Graham Wakefield (Eds.). nime.org, 287–290. http://nime.org/proceedings/2013/nime2013_232.pdf

[6] John C. Hart. 1996. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *Visual Computer* 12, 10 (1 Jan. 1996), 527–545. https://doi.org/10.1007/s003710050084

[7] Timo Hoogland. 2020. *wave terrain synthesis*. https://github.com/tmhglnd/wave-terrain-synthesis

[8] Couturier Jean-Michel. 2002. A Scanned Synthesis Virtual Instrument. In *Proceedings of the 2002 Conference on New Interfaces for Musical Expression* (Dublin, Ireland) *(NIME '02)*. National University of Singapore, SGP, 1–3.

[9] Riccardo Marogna. 2018. CABOTO: A Graphic-Based Interactive System for Composing and Performing Electronic Music. In *18th International Conference on New Interfaces for Musical Expression, NIME 2018, Blacksburg, VA, USA, June 3-6, 2018*. nime.org, 37–42. http://www.nime.org/proceedings/2018/nime2018_paper0010.pdf

[10] mediamolcule. 2021. *Dreams*. http://dreams.mediamolecule.com/

[11] Eduardo Reck Miranda. 1995. Sound design : an artificial intelligence approach.

[12] Andrew J. Oxenham. 2012. Pitch Perception. *Journal of Neuroscience* 32, 39 (2012), 13335–13338. https://doi.org/10.1523/JNEUROSCI.3815-12.2012 arXiv:https://www.jneurosci.org/content/32/39/13335.full.pdf

[13] Joshua Peschke and Axel Berndt. 2017. The Geometric Oscillator: Sound Synthesis with Cyclic Shapes. In *Proceedings of the 12th International Audio Mostly Conference on Augmented and Participatory Sound and Music Experiences* (London, United Kingdom) *(AM '17)*. Association for Computing Machinery, New York, NY, USA, Article 15, 6 pages. https://doi.org/10.1145/3123514.3123522

[14] Les Piegl and Wayne Tiller. 1997. *The NURBS Book (2nd Ed.)*. Springer-Verlag, Berlin, Heidelberg.

[15] William J. Poser. 1990. Douglas O'Shaughnessy, Speech Communication: Human and Machine. Reading, Massachusetts: Addison-Wesley Publishing Company, 1987. Pp. xviii 568. ISBN 0-201-16520-1. *Journal of the International Phonetic Association* 20, 2 (1990), 52–54. https://doi.org/10.1017/S002510030000431X

[16] Frederic Anthony Robinson. 2021. Debris: A playful interface for direct manipulation of audio waveforms. In *NIME 2021*. https://nime.pubpub.org/pub/xn761337.

[17] Allan Seago, Simon Holland, and Paul Mulholland. 2004. A Critical Analysis of Synthesizer User Interfaces for Timbre. In *Proceedings of the XVIII British HCI Group Annual Conference HCI 2004*, Andy Dearden and Leon Watt (Eds.). Vol. 2. Research Press International, Bristol, UK, 105–108. http://oro.open.ac.uk/5688/

[18] Keisuke Shiro, Ryotaro Miura, Changyo Han, and Jun Rekimoto. 2019. An Intuitive Interface for Digital Synthesizer by Pseudo-Intention Learning. In *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound* (Nottingham, United Kingdom) *(AM'19)*. Association for Computing Machinery, New York, NY, USA, 39–44. https://doi.org/10.1145/3356590.3356598

[19] S. S. Stevens, J. Volkmann, and E. B. Newman. 1937. A Scale for the Measurement of the Psychological Magnitude Pitch. *The Journal of the Acoustical Society of America* 8, 3 (1937), 185–190. https://doi.org/10.1121/1.1915893 arXiv:https://doi.org/10.1121/1.1915893

[20] Anil Çamci. 2018. Graintrain: A Hand-Drawn Multi-Touch Interface For Granular Synthesis. (Jun 2018). https://doi.org/10.5281/zenodo.1302528