A Critical Analysis of Contemporary Graphic Sound Synthesis Software

Raphael Radna
rradna@ucsb.edu
Dept. of Media Arts & Technology
University of California Santa Barbara

ABSTRACT
This paper presents an analysis of five contemporary software applications in order to assess the current state of graphic sound synthesis: MetaSynth, HighC, Virtual ANS, UPISketch, and SketchSynth. Focusing on the mapping strategies employed between image and sound, I seek to elucidate the way the features, interfaces, and interaction methods of these programs affect their expressivity in the visual and musical domains, with the goal of identifying opportunity spaces for the creation of new software.

1. INTRODUCTION

Graphic sound synthesis describes a mechanical or computational process by which visual media (usually a painting, drawing, or photograph) is transformed into electronic music. This multi-modal and interdisciplinary practice has been a subject of inquiry amongst experimental artists, composers, and inventors for more than a hundred years, in this time yielding numerous projects that have realized it through varied technological interventions. While each implementation offers an idiosyncratic approach, certain design paradigms have arisen and achieved near-universality, namely the pitch-versus-time (spectrographic) interpretation of the 2D image. The staying power of this interface is partially cultural, but may be owed at least in part to its use in the UPIC, an influential digital graphic sound synthesis system first developed by composer Iannis Xenakis in the 1970s [1].

The present study probes a range of current graphic sound synthesis applications to determine what interactions and mappings between visual and sonic forms they employ or support. I am concerned with the relative expressivity of the applications in the visual and sonic domains, taking this as a measure of suitability of the chosen mapping and accompanying feature set. A secondary area of interest is the user interface, and the ways in which it supports creative expression or causes friction with the mapping paradigm embodied by the software. Through this analysis, I aim to elucidate the attributes that make a graphic sound synthesis application successful, and also to discover directions for new designs not currently explored by available offerings in the field. I am particularly motivated by potential alternatives or augmentations to the spectrographic model of representation, which is deeply embedded in graphic sound synthesis.

The applications selected for this analysis include MetaSynth, a long-standing music composition and sound design workstation built around a robust graphic sound synthesis engine [2]; HighC, a music composition program centered on graphic control of curves representing individual sonorities of continuously-variable pitch [3]; Virtual ANS, a modern digital remake of the unique ANS optical synthesizer developed in the mid-century Soviet Union [4]; UPISketch, a contemporary reinvention and extension of the UPIC idea [5]; and SketchSynth, a prototype for graphic sound synthesis intended for the sonification of manual drawings [6]. Because I seek design directions for new software, I made the decision to limit my analysis to recent applications. I defined these as software that is contemporary, compatible, and available; all selections are currently maintained (with the possible exception of HighC, which appears to have ceased development in 2012 [7]), available to download from the internet as either free software or as a trial version, and compatible with modern systems. Since my analysis was conducted through personal usage of the selected tools, these considerations were also practical; while numerous and diverse systems exist, only a subset was available for evaluation. A few of these applications are available on several platforms, including versions for touchscreen mobile devices (Virtual ANS, UPISketch); for consistency, all analysis was conducted using the desktop versions of the software.

2. BACKGROUND

Graphic sound synthesis is motivated by the desire to generate electronic sound according to existing vocabularies of visual art and design. It is a rare artistic process in that it produces two artifacts: an image and

a sound recording. As a compositional tool it offers unique affordances, and frequently results in sounds that would be difficult or impossible to produce by conventional means. It can also be used to create visual art exclusively or in performance. Besides experimental composers, it also appeals to adventuresome artists intrigued by the problem of image sonification, and non-specialist amateurs interested in experimenting with interactive technology and both creative modalities. The immediacy of drawing as an interface makes graphic sound synthesis appropriate for pedagogical settings involving children, an aspect recognized and used during the UPIC days [8]. Systems design for graphic sound synthesis remains an open question, owing to the fundamental differences between visual and sonic media [9].

Graphic sound synthesis requires the translation of visual media, inherently physical and static, into an audible form which is essentially intangible and transient. This is a non-trivial and open-ended problem; indeed, there are any number of ways in which an image may be transformed into sound. With no definitive solution, developing graphic sound synthesis software requires the elaboration of a mapping strategy for converting visual data into audio samples. A popular approach is to interpret an image as a spectrogram, a graph that plots the amplitude of individual frequency components of a sound over time, ordinarily obtained by applying the short-time Fourier transform to a digital sound file. Sounds represented in this fashion can be synthesized using additive synthesis, or the inverse fast Fourier transform. This paradigm borrows from deeply-embedded spatial metaphors for the graphical representation of data, most directly those employed by traditional Western music notation, which represents pitch using height, and time as progressing to the right.

Fig. 1. A spectrogram of a flute tone. Frequency is on the y axis, time is on the x axis, and amplitude is represented by the darkness of the image.

In contrast to a musical score, which presents selective symbolic performance data, a spectrogram conveys a complete and precise physical specification of a sound. Since the spectrogram is exact in its acoustic description, yet conceptually distant from human perception of sound, it is problematic as a compositional interface. While a musician can infer details left out of a score, the digital synthesis of a spectrogram allows no room for interpretation. It follows that a simple score can result in complex sounds, but the same is never true for a spectrogram; it is difficult to create elaborate, evolving timbres using this paradigm. The result is that music made using a spectrographic interface is often characterized by simple timbres which fatigue the ear.

Drawing music as a spectrogram requires one to simultaneously design sounds, their changes in timbre over time as variations in the amplitude of individual frequency components, and musical form at all timescales, an exceedingly daunting prospect even before including high-level considerations such as dynamic balance and spatial trajectories. Furthermore, it requires these details to be committed concretely, while research shows that early stages of composition rely on vagueness and ambiguity [10]. A related difficulty is presented by the fact that spectrograms use an absolute time scale, while traditional music composition deals with measures of time relative to a reference duration; however, this feature is shared by other contemporary composition methods, and may therefore be more readily assimilated.

As much as the spectrographic interface presents a challenge for composition, it also constrains visual design possibilities. Its consequences, for example that a shape or gesture in the upper left corner of an image should occur first and correlate with high-pitched sound, are arbitrary. Such a mapping disregards the components of an image and their relation to its overall form, which is on its own timeless; it simply occurs

when we choose to treat it as a score or spectrogram, placing the visual composition in a position of secondary importance.

Many such graphic sound synthesis systems have been developed with the goal of enabling a scientific specification of music; these are visual composing applications, and their graphical output is essentially a data visualization. Is it possible to imagine a paradigm that strikes a balance with what we might call musical drawing, enabling comparable expressivity in the artistic domain without compromising, and even expanding, what is possible musically? Such a system would more closely align the goals of the visual and musical arts, and constitute a more complete vision of graphic sound synthesis. While this is admittedly a difficult path, this research represents an initial attempt to help point in its direction.

3. METHODOLOGY

The five applications included in this analysis were evaluated after a preliminary period of use, during which I attempted to learn the software well enough to produce an artifact that is representative of its capabilities in the hands of a casual user. Due to the disparities in relative complexity among the applications, this period was not a measured duration of time, but was based on a self-assessment of my comfort with the interface; the evaluations lasted around 1 to 2 hours on average.

Before beginning the evaluation, I developed guidelines for analysis addressing several dimensions of comparison intended to direct my usage of the software and help organize my thoughts during the process. These dimensions addressed the facility of graphic interaction; quality of artistic expression; quality of musical expression; sense of correlation between graphic and sonic elements; customization of mappings; novelty of interface; accessibility to novices; support of compositional precision; similarity of interaction to established digital art paradigms; and similarity of interaction to established digital music paradigms.

After completing the work with the software, I assigned each a numerical rating for artistic and musical expressivity according to a set of discrete criteria drawn from the aforementioned dimensions. These ratings took the form of a five-point Likert scale, and thus best measure the relative performance of the selected tools. The same dimensions and scale were used for all software. The results were used to visualize the artistic and musical expressivity of the applications.

Evaluation Criteria for Artistic Expression
(1=Strongly Disagree, 2=Disagree, 3=Neither Agree or Disagree, 4=Agree, 5=Strongly Agree)

	MetaSynth	HighC	Virtual ANS	UPISketch	SketchSynth
Enables a multiplicity of artistic styles	4	1	2	1	5
Adheres to established digital art interaction paradigms	4	3	4	3	1
Interface supports fast workflows for artistic creation	3	4	3	2	1
Enables artistic expression for non-experts	4	3	4	3	2
Facilitates precise work in the visual domain	4	5	4	4	1
A wide variety of visual effects are possible	5	1	4	1	5
Contributes novel interaction modalities for authoring visual media digitally	4	2	3	2	1

Evaluation Criteria for Musical Expression

	MetaSynth	HighC	Virtual ANS	UPISketch	SketchSynth
Enables a multiplicity of musical styles	4	5	3	5	2
Adheres to established digital music interaction paradigms	4	5	5	4	3
Interface supports fast workflows for musical creation	3	3	2	2	2
Enables musical expression for non-experts	5	4	4	5	4
Facilitates precise work in the audio domain	4	4	3	4	1
A wide variety of sonic effects are possible	5	4	2	4	3
Contributes novel interaction modalities for authoring audio media digitally	5	2	1	4	4

Fig. 2. Tabulation of artistic and musical expression of each program according to discrete criteria.

4. EVALUATION

The following sub-sections contain narrative accounts of observations made during the evaluation of each software.

4.1. MetaSynth

MetaSynth is a full-featured music composition and sound design program based on graphic sound synthesis. Developed by Eric Wenger and first released in 1998 [11], it is a regularly-maintained commercial application; the trial version was used for this study. MetaSynth offers six separate but related interfaces that enable different modalities for graphical interaction with sound, offering functionality such as drawing wavetable oscillators, sequencing patterns of notes, and arranging samples and sequences in a multi-track editing environment. The component of the software used for this analysis is the "Image Synth," which provides a canvas for the user to draw a sonographic representation of sound.

The Image Synth features a relatively small canvas, with a maximum width of 4,096 and height of 1,024 pixels. The pitch range of the sonification is determined by the canvas height, a reference tone, and a musical scale. The range is not normalized: for example, given the same canvas height and reference a tone, a chromatic scale will have a higher maximum pitch than a scale with 32 equal divisions of the octave. This permits the user to select settings that place portions of the canvas into the infrasonic and ultrasonic ranges.

As a practicality, the possible height resolution is constrained by the number of synth voices that can be rendered simultaneously in real time, but the width restriction seems unnecessary. However, the timescale is only partially determined by the canvas width; the maximum width of 4,096 pixels and default tempo of 120 beats per minute supports a duration of 1 minute and 35 seconds, but longer durations could be achieved by reducing the tempo, at the cost of time granularity. The ability to zoom the interface makes it possible to work intricately at small time scales.

Visually, MetaSynth uses greyscale to represent mono sound, and color for stereo; rotation of hue between maximum red and green values determines position in the stereo field. Gradients can be applied to create spatial movement. In either case, the brightness of a pixel determines the amplitude of its corresponding oscillator. The blue channel is reserved for a silent layer, which can be used to draw or import custom pitch and time reference grids which are ignored by the sonification. This regime works well, but

limits expressivity in the visual domain, due to the greatly-reduced color palette. MetaSynth supports layers, but only the active layer is displayed in full color.

The Image Synth provides a nice variety of brushes; in addition to some preset geometric shapes, curves, and a basic freehand tool, some of the creative additions include an airbrush tool, which creates marks with soft edges; a harmonic drawing tool, which creates parallel marks up to the twelfth overtone above the cursor; a gradient drawing tool, which creates shaded regions; a spray paint tool, useful for coloring a region stochastically; and three different smearing tools, which affect previously-colored pixels and their neighbors in various ways. All marks can be applied as dots, lines, or repeated points at a definable step, which is useful for creating recurring patterns. A large number of visual filters and effects can be applied, opening up many creative possibilities in the visual domain, as does the ability to import and modify images from files.

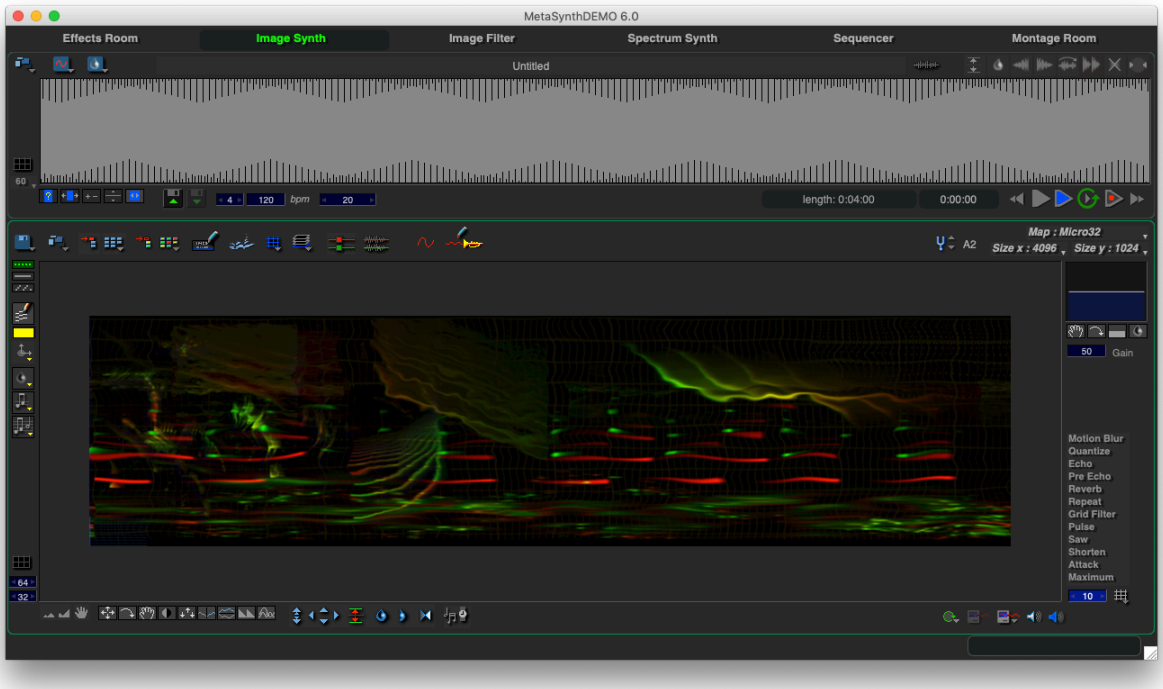

Fig. 3. MetaSynth's "Image Synth" interface. The prominent black rectangle is the canvas, surrounded by GUI toolbars related to drawing, applying visual effects and filters, selecting the canvas size, specifying instrument settings for sonification, and more.

The main drawbacks of using the Image Synth compositionally are its lack of per-gesture control, and that it offers only one undo step. Since there is no way to group marks into logical units, the workflow breaks down at the mesostructural level of composition. This can be mitigated somewhat by creating a new layer for each gesture, but this solution quickly becomes cumbersome if many layers are used. The limited undo functionality discourages exploration, which is a pity for a program with so many deep and interesting features. Furthermore, some operations don't register in the undo queue, which in my experience frequently lead to unexpected loss of work.

However, the sound engine used in the Image Synth is robust, and offers lots of possibilities for determining how the canvas is sonified. While it is traditional to use sine waves in additive synthesis, here we are able to create and choose complex waveforms, or sample-based sound generators, which can produce richer sonic results. It is important to note, however, that only sine waves render the image faithfully as a spectrum; sources with complex spectra will quickly tend towards noise given high-density graphical input.

4.2 HighC

HighC is a visual music composition program developed by Thomas Baudel and first released in 2007. Available in free and licensed versions, its stated goal is "to make music composition as simple and direct as sketching [3]." HighC takes inspiration from the UPIC in that it regards gestures as individual musical events with variable pitch. It uses a vector representation that enables gestures to be cut, copied, deleted, moved, scaled, stretched, and fully edited after creation, down to the addition, deletion, or adjustment of individual points. Its selection tool allows multiple gestures to be moved or resized as a unit. These features support workflows in which compound gestures are created from multiple curves, including on short time scales. HighC uses a JSON-style representation, making it possible to generate scores using a scripting language.

However, the visual expressivity of the program is limited; besides drawing the curves, all the user has available is to change their color and stroke geometry by selecting a waveform and amplitude envelope, respectively. This lends some visual interest to the score, but since the color corresponding to each waveform is arbitrary, the mapping is not immediately comprehensible [12]. The curves are rendered in an attractive calligraphic style, which depicts the amplitude envelope applied. This introduces variation among the curves, but relegates visual style to an indicator of sonification.

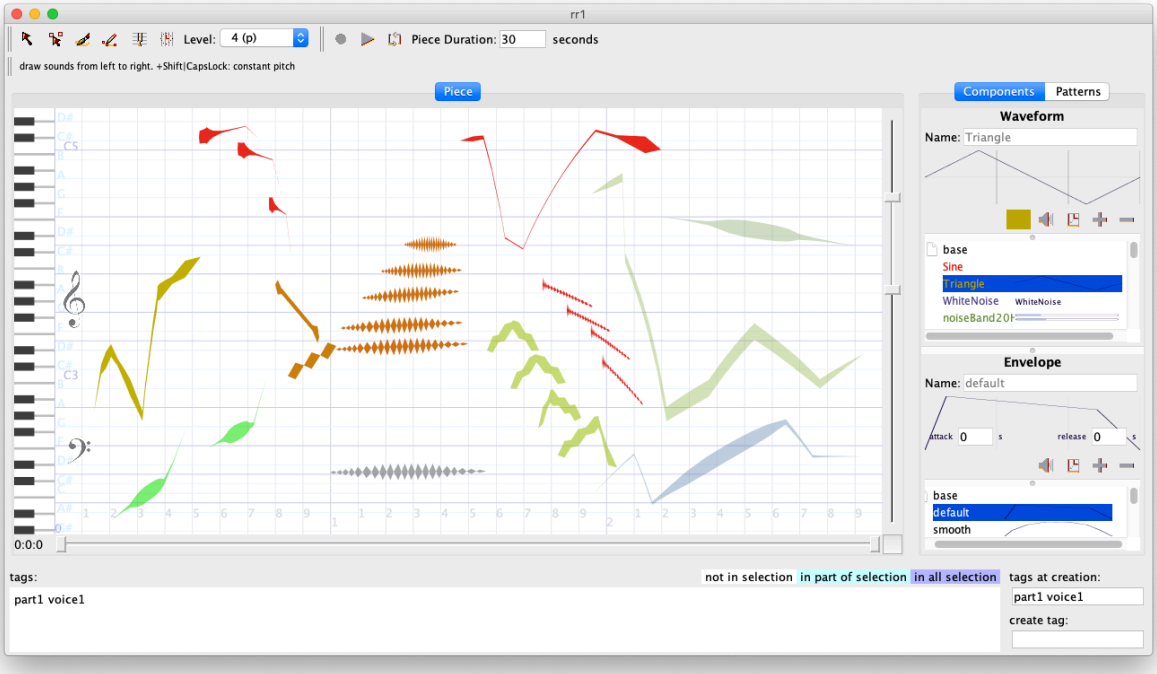

Fig. 4. HighC interface. Several gestures are shown on the canvas. Their color is determined by waveform, and their shape by amplitude envelope. Both of these are selected from the menus to the right side of the window.

Through the ability to select waveforms and envelopes on a per-curve basis, the synthesis back-end of HighC enables the realization of multiple sounds from a single canvas. The included sounds are somewhat plain and of varied quality; the full version of the program enables the use of external audio samples. It also allows the user to specify custom waveforms from a GUI interface, but doing so is not straight forward. The ability to apply frequency and amplitude modulation presents a possibility for creating more complex timbres, however in a very characteristic way.

In my view, HighC's functionality as an electronic music sequencer has some oversights, such as the way dynamics are handled. Dynamics can be assigned at the gesture level, and correspond visually to its alpha level, but only using a scale of integer values from 0 to 9, which is a very coarse resolution for music. Additionally, this value can only be applied to a gesture as a whole, rather than as a time-varying parameter.

This can be mitigated by the use of envelopes, but this would force the user to manage many envelopes that are identical other than their maximum level. HighC does not provide control over stereo panning.

4.3 Virtual ANS

Virtual ANS is a graphic sound synthesis application inspired by the original ANS photoelectronic sound synthesizer. Developed since 2007 by Alexander Zolotov, the desktop version is available for free from his website. Composing with the original ANS synthesizer was done by etching shapes into a layer of mastic on a glass surface representing a spectrogram [13]. Virtual ANS functions similarly, but supports up to 1,440 sinusoidal oscillators, a range of digital graphics manipulation facilities, and bilateral conversion between sound and image. Like HighC, it displays a piano roll-style pitch reference to the left side of the canvas (the original ANS did as well), and like MetaSynth, it uses a raster drawing paradigm. Its canvas can be set to arbitrary resolutions, including very large dimensions. Color is represented using a rainbow palette across the entire pitch range selected for the project; while the result is beautiful, it does constrain visual expression by enforcing a strict correspondence between hue and pitch. The mapping can be customized to an extent; the colors can be flipped, darkened, brightened, or made to repeat at an integer subdivision of the full pitch range. The display renders attractive lighting effects where the play head intersects a gesture on the canvas.

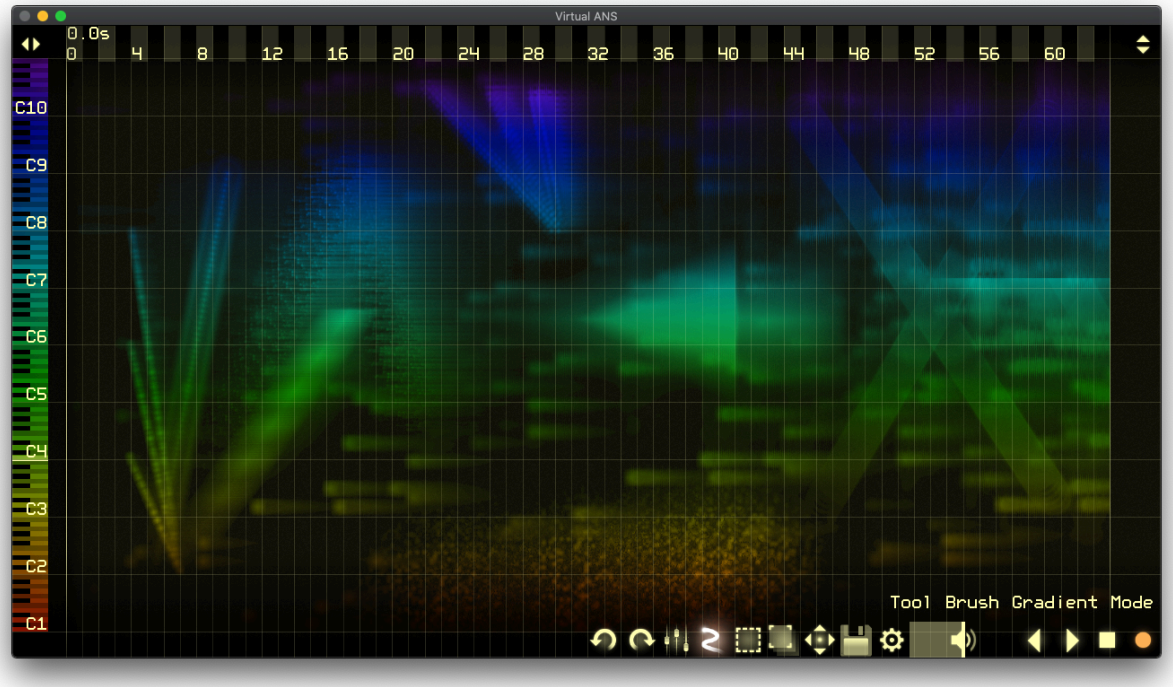

Fig. 5. Virtual ANS interface. The canvas uses a partially-customizable uniform color palette, with hue bound to pitch.

The software offers fourteen brush shapes, ranging from fine to diffuse marks, including spray paint effects and textured patterns. Additional brush textures can be imported. These can all be scaled, and their spacing altered to adjust between continuous and discrete drawing when the mouse is dragged. In addition to freehand drawing, line, triangle, and rectangle methods are available, using either a solid fill or various gradients. Draw modes such as lighten, add, subtract, divide, and exclusive-or contribute some complexity to what can be accomplished; some of these allow the user to draw negative space into their designs, only taking effect on regions where marks had previously been made. Like MetaSynth's smearing tools, this exploits the bitmap representation of the canvas to create complex forms that would be difficult to achieve otherwise.

Virtual ANS also allows the user to select a rectangular region to copy, cut, and paste. However, it lacks the ability to translate, resize, or rotate a region by dragging the mouse, an element of direct manipulation

from which it would benefit. As in MetaSynth, the bitmap representation makes it difficult to control which pixels are included in a selection; additional layers can be made to mitigate this problem. A number of visual effects can be applied to a selection or to an entire layer, which greatly augment the visual and sonic possibilities of the program, including vertical and horizontal blurring, contrast adjustment, gamma correction, and edge detection, to name a few.

Virtual ANS provides up to 16 undo steps, which is high enough to encourage safe experimentation with software features. While the interface is attractive and compact, it has many similar-looking menus; in my experience, the monotony made it easy to miss important features. The sonification component of Virtual ANS is limited to additive synthesis with sine waves. The desired pitch range can be specified from 1 to 12 octaves, in up to 120 equal divisions of the octave: a pitch resolution of 10 cents. The user can also set the base frequency, beats per minute, and pixels per beat (time resolution) of the project.

4.4 UPISketch

UPISketch is a recent software project intended to recreate and expand the functionality of the UPIC system. It was developed by a team lead by Rodolphe Bourotte, supported by the Centre Iannis Xenakis and European University of Cyprus, and first released in 2018 as free software. Like the UPIC, UPISketch treats each individual mark as a single sonic event with continuously-variable pitch and amplitude. To accomplish this, it uses a vector representation of the canvas, enabling the user to move gestures after drawing, as well as edit the individual points of a curve. UPISketch stores projects in a human-readable format that organizes gestures as structures containing pitch and amplitude tags, containing coordinates for time and value.

The canvas is a grid of pitch and time with unit markers on the axes, but it is endlessly extensible in the horizontal dimension and zoomable in the vertical dimension, a surprisingly under-utilized affordance of the digital medium. Gestures are created by selecting a sound sample from a library, and then clicking and dragging on the canvas with the mouse. Once a gesture is drawn, a bounding box is created around it; clicking on this enables the user to edit the previously-drawn red pitch curve, as well as to apply a loudness contour by adding and modifying nodes on a blue amplitude curve. While this enables precise manipulation of fine details, when many gestures are drawn in close proximity, it can become impossible to click on a specific one to modify it, as it becomes occluded by the others.

The visual component of the software is not highly aesthetic, and as in HighC, was designed more as a functional notation system than as a vehicle for artistic expression in its own right. Other than drawing, modifying, and placing the gestures themselves, all the user has to affect their appearance is to change the color of their translucent bounding box by combining them into groups. This again serves a primarily pragmatic function; grouped gestures can be moved as a unit. In my work with the software, I grouped all gestures of the same instrument, in order to visualize the orchestration.

While UPISketch provides a minimalistic interface, it lacks some standard computer interactions that would improve workflow. For example, it is not possible to nudge a selected gesture with the arrow keys as one might expect. Likewise, the application doesn't allow the user to copy and paste pitch or amplitude envelopes. This slowed my work with the program, and was compounded by the strictness with which it enforces its different mouse modes, selected using icons on the toolbar at the top left of the interface.

UPISketch's synthesis engine is based on sample playback; it contains a default sample library featuring orchestral instruments and sound effects, but user samples can also be loaded or recorded directly from the interface. Sonification uses a sophisticated algorithm which performs pitch estimation and granular synthesis to allow playback of audio samples at arbitrary transpositions and time adjustments [14]. This is an automated process; all parameters are hidden, with nothing for the user to tune. The implementation is impressive, but in certain contexts, the resulting sounds appear noticeably artificial. While I admire that UPISketch conceals this complexity from the user, some ability to modify the synthesis could make it a more expressive instrument. By only offering control over pitch and amplitude, many elements of electronic music are neglected, including spatialization, effects, and expressive inflections such as vibrato. The next update of UPISketch will offer more control over the synthesis backend [15], which I suspect will address these concerns. As a compositional tool, it would also benefit from higher-level controls (for example, the ability to set an overall level for all gestures of the same source).

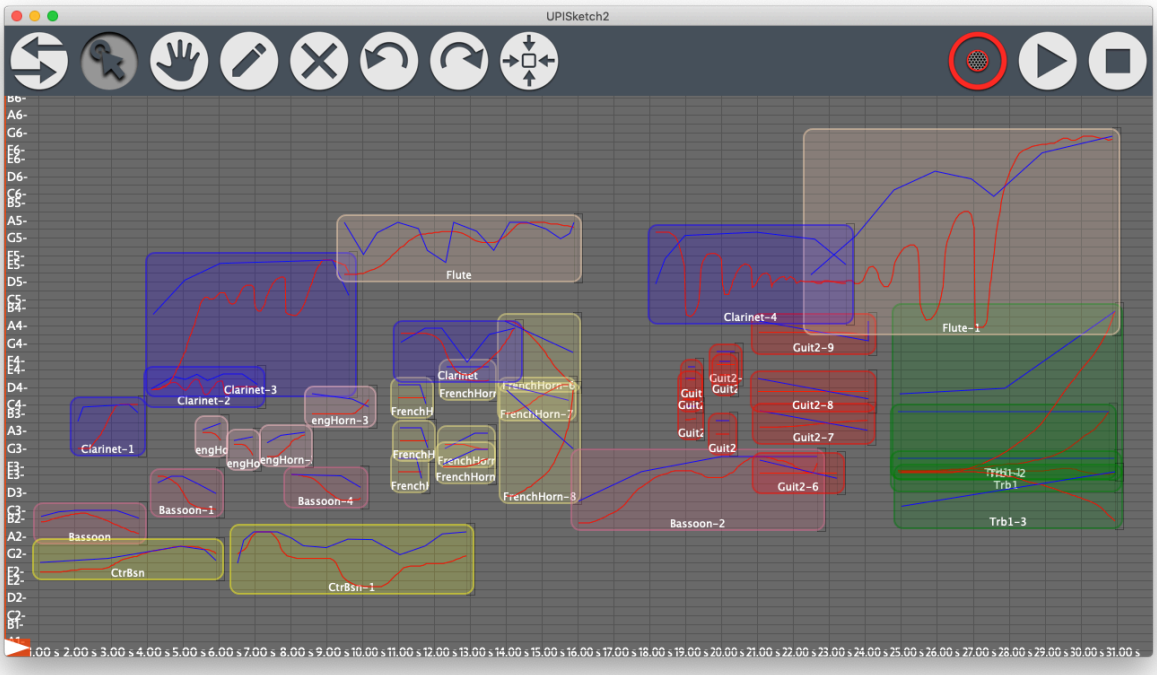

Fig. 6. UPISketch interface. Several gestures are drawn, each with pitch and amplitude curves displayed. They have been grouped by instrument, which is reflected in the bounding box color.

4.5 SketchSynth

SketchSynth is a new project in the graphic sound synthesis space. Developed by artist JeongHo Park, it was first released in 2020 as open source software. It is unlike any of the other projects considered in this analysis in that it outsources the drawing component, operating only on imported images. While this potentially maximizes artistic expression by keeping the visual design element in the physical domain, it also requires traditional artistic ability and access to materials; the lack of digital drawing affordances might discourage amateur or experimental users. Any bitmap image file can be loaded, but the developer intends for hand-drawn images to be used; the program starts up with an image featuring several designs scribbled in a weekly planner, giving the program a decidedly "AR" aesthetic.

SketchSynth's interface presents a large view of the imported image, with red and green shaded regions that can be moved with the mouse. The green region defines the sonification area, while the red one is interpreted as a wavetable used by the oscillator bank, defining the timbre of the synthesis voices. While this is of limited use for deterministic expression due to the difficulty of anticipating timbre from a time-domain waveform [16], it is a unique feature that lends the program a certain playfulness and charm. The system thresholds the pixels under both the play and wavetable regions to binary representations, which determines when to play and which samples to read into the oscillator buffer respectively.

The downside of this approach is that it doesn't handle brightness or color; the binary rendition means that any oscillator can only ever be on or off. This leaves no opportunity for variable dynamics, panning, or other parameters of the sonification to be specified graphically. External GUI controls enable the user to retune the pitch range of the play area, adjust the payback rate, and control an overlap setting, which works like an envelope control by adding a release tail to the activated oscillators. The GUI also provides contrast controls for each thresholding process, affecting the way SketchSynth interprets the play and wavetable regions. These can be modified during playback, which adds some performativity to the program.

SketchSynth is clearly in an early phase of development, but already exhibits some compelling original ideas. It could be improved by the addition of multiple play and wavetable regions, and the ability to resize them. A video posted by the developer demonstrates functionality whereby multiple play regions are strung

together in a sequence, an alternative to the standard interpretation of time inherited from the spectrographic representation [17]. Allowing the user to control the flow of time by defining paths through the image can help liberate visual form by decoupling graphic location from sonic temporality.

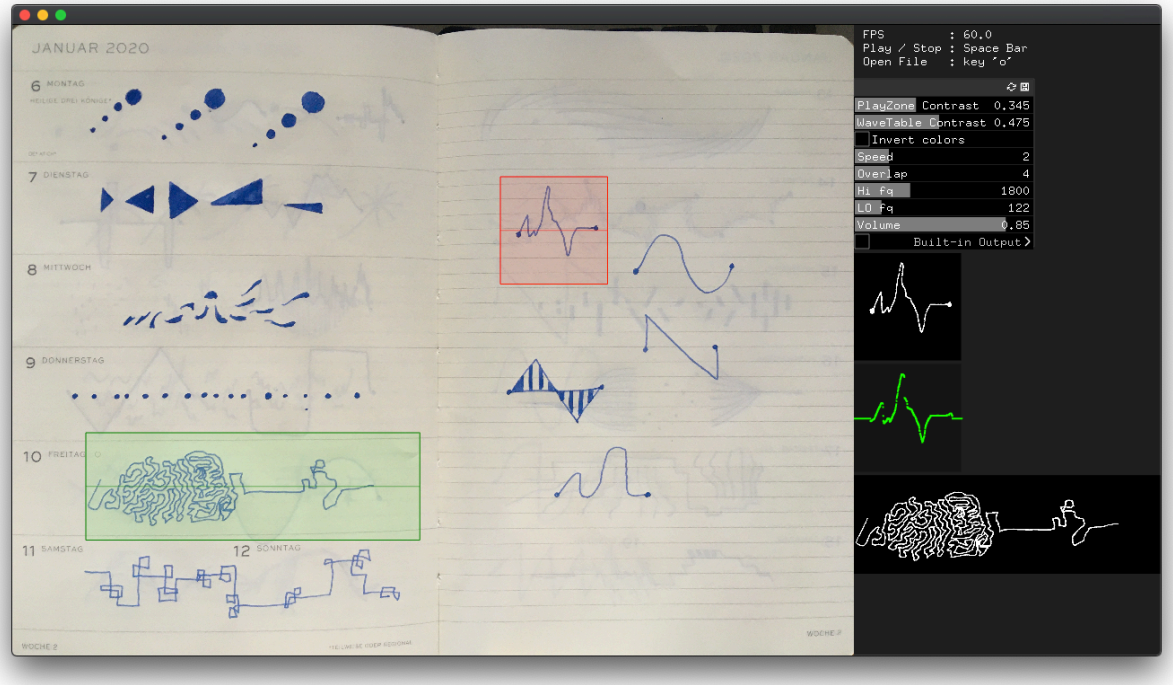

Fig. 7. SketchSynth interface. Note the green playback and red wavetable regions, and the viewfinders that preview the binary rendition using current contrast values in the right pane.

5. DISCUSSION

While each of the five applications evaluated presents a graphic interface representing pitch as a function of time, they sort fairly easily into one of two sonification regimes: they either treat each row of the canvas as a discrete sound generator, or each gesture as a sound to be triggered and modulated according to its placement and contour. These sonification strategies are linked to graphical representation modes, with the former requiring a raster representation and the latter a vector one.

As can be seen in the figure, the digital-drawing applications that employ a raster paradigm (MetaSynth, Virtual ANS) were rated higher on the measure of artistic expression. This is because the contextual processes enabled by conceiving of the canvas as an array of pixels makes possible visually-complex drawing and filtering operations, resulting in more varied and attractive designs. These programs also use color more expressively than their vector counterparts, though only MetaSynth maps it to a property of the sonification. The compromise is that, while these programs allow immediate and intuitive exploration with graphic sound synthesis, they do not support high-level compositional processes very well; grouping gestures is only achieved by using layers, a simple approach that breaks down as complexity increases. MetaSynth's Image Synth also places limits on canvas size, making it difficult to author large-scale works, and while it provides a very flexible audio engine, it, Virtual ANS, and SketchSynth suffer from a limitation shared by the bitmap-oriented approaches in this study: they use a single instrument to sonify the canvas.

In contrast, the vector applications (HighC, UPISketch) generally scored highly in the measure of musical expression (though they were surpassed by MetaSynth, on account of its robust synthesis framework). The per-gesture representation of sound imparts many advantages, such as allowing the user to orchestrate the canvas using a variety of sound sources. It enables powerful and precise interaction with the canvas in all dimensions and at all time scales; at the same time, these programs do not offer any capabilities for processing the canvas visually. This results in less expressive graphics that use symbolic representation to

convey data clearly, but do not support a wide variety of artistic styles. Additionally, although the pitch-over-time representation is maintained, the spectrum is no longer visualized, so the audiovisual mapping becomes increasingly abstract.

While these programs provide more precise control, they tend to support the specification of fewer sonification parameters: in both cases, only sound source, pitch, and amplitude are accessible. To be sure, this is an area where the raster applications could be improved as well; however, that canvas paradigm and the operations it enables inherently supports more diverse and time-variant expression. I was surprised to find that, among the evaluated applications, only MetaSynth included provisions for controlling panning, a fundamental element of electronic music.

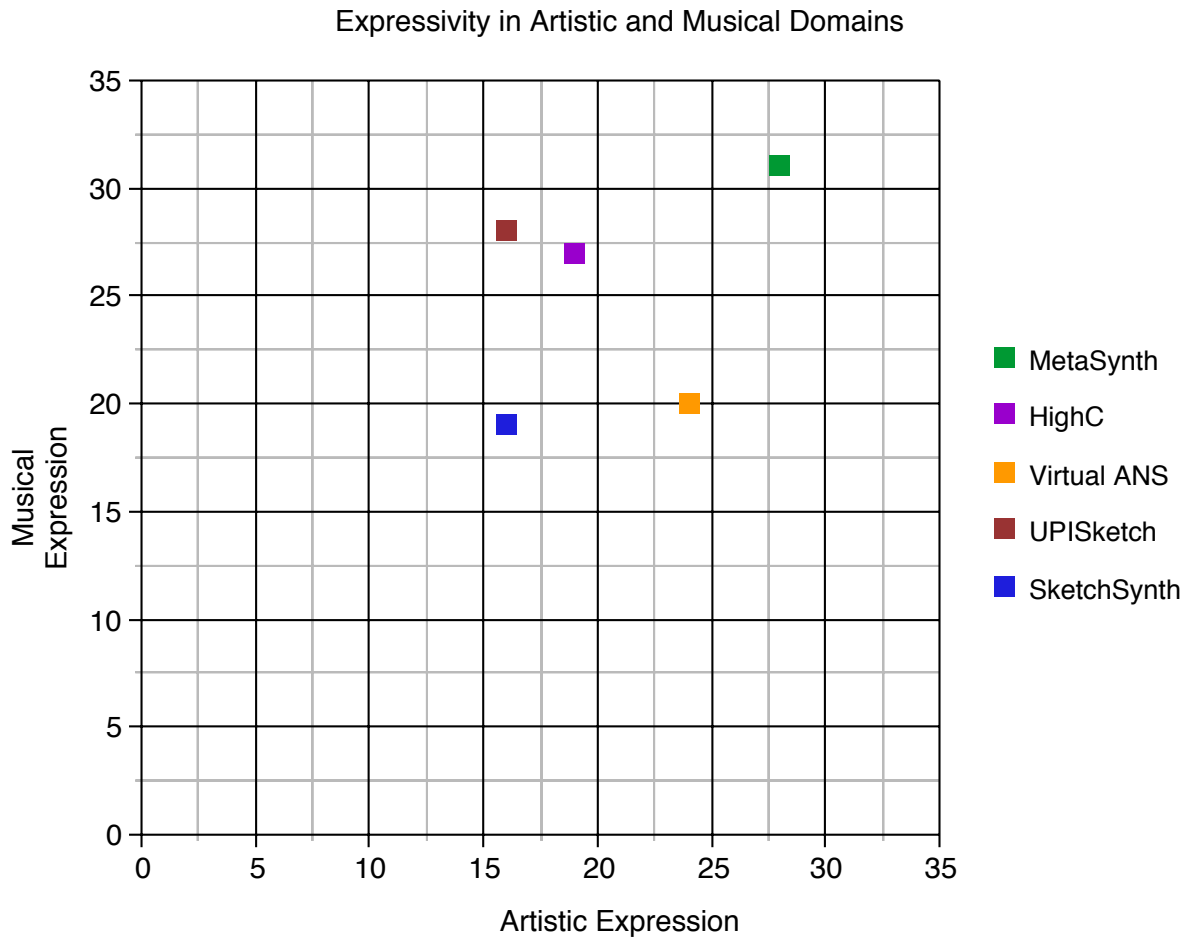

Fig. 8. A plot of musical and artistic expression derived from the software evaluations.

In light of this analysis, I propose the following guidelines for the creation of new graphic sound synthesis software:

- Implement a canvas that is resizable to arbitrary dimensions and endlessly zoomable, enabling composition at all durations and time scales.
- Allow drawing with all possible colors, and convey sensorially meaningful mappings to the sonification engine.
- Offer a procedure for grouping and operating on gestures at a high level, while still displaying spectra and allowing graphical operations on them to a fine degree of granularity, and at all time scales.
- Support fast workflows by incorporating maximum direct manipulation of the canvas and its contents, contextual mode switching, keyboard shortcuts, and adherence to established general and domain-specific interaction metaphors where possible.

- Register all actions that affect the canvas area in an undo history that extends to the beginning of the program session.
- Enable sonification with multiple timbres using one program instance, either via layers, per-gesture control, multiple canvases, or some other novel interface element.
- Provide an infallible method of defining the pitch scale and range of the sonification; all graphical elements on the canvas should be sonified regardless of this setting.
- Make possible continuous control over spatialization and other sonification parameters. This is an expansive and ill-defined space; user control over mapping definitions is likely required.

Since the affordances and constraints of the raster and vector representations are largely complimentary, a hybrid approach that allows high precision and per-gesture control while visualizing and permitting graphical operations on individual spectra is indicated. Admittedly, this direction merely extends the spectrographic paradigm, albeit in a way that ameliorates its faults. Due to the relative similarity of the software considered, it is impossible to make a judgement about interfaces that abandon a spectrographic representation entirely, those that enable full user specification of audiovisual mappings, or those that provide alternative conceptions of time. These are the most radical areas for the development of new paradigms in graphic sound synthesis, and potentially the most fertile. Unfortunately, though such projects are known to exist [9, 11], none of them met the selection criteria for inclusion in this study. Future research that looks at such projects would be necessary before making a statement on the current viability of these approaches.

6. CONCLUSION

Graphic sound synthesis is an active area of research in digital art, music, and human-computer interaction. In this study, I analyzed five contemporary applications in this domain, in order to summarize the available modalities, to evaluate effectiveness and expressiveness of individual implementations, and to highlight opportunities for further development. Towards this aim, I documented my usage of each application, applying my observations to better understand the merits of each for artistic and musical expression. The results indicate that the strengths of each application were correlated with the representation used for the graphical interaction area: while raster approaches generally enabled more expressive visuals, vector interaction facilitates more sophisticated musical constructs. Accordingly, I proposed a set of guidelines for the development of new software that might leverage the strengths of each paradigm.

Some challenges of this study arose from the disparities amongst the selected software. As a sample, these programs vary widely in development age; type and availability of documentation, tutorials, reference material, and research publications; and online presence, including user communities. These factors affect both the maturity of the application's features and my ability to become acquainted with them in a short period. Another weakness of the analytical approach came from the difficulty of standardizing usage across tools with different interfaces and affordances. A study of a more expansive scope could mitigate this limitation by surveying existing user bases for each application. Such an approach would accrue the benefits of representing perspectives and workflows of practitioners from other fields of expertise, with varied artistic backgrounds and technological inclinations. I am particularly interested in how visual artists might assess these applications.

REFERENCES

1. Marino, G., Serra, M.-H., & Raczinski, J.-M. (1993). The UPIC System: Origins and Innovations. Perspectives of New Music, 31(1), 258–269. doi: 10.2307/833053

2. *MetaSynth 6*. (n.d.). http://www.uisoftware.com/MetaSynth/

3. *Draw your music*. (2020, March). HighC. Retrieved March 21, 2020, from https://highc.org/

4. *Virtual ANS Spectral Synthesizer*. (2020, March 21). WarmPlace.ru. Retrieved March 21, 2020, from https://warmplace.ru/soft/ans/

5. *UPISketch*. (n.d.). Centre Iannis Xenakis. Retrieved March 21, 2020, from http://www.centre-iannis-xenakis.org/upisketch

6. *jeonghopark/SketchSynth-Simple*. (2020, February 11). GitHub. Retrieved March 21, 2020, from https://github.com/jeonghopark/SketchSynth-Simple

7. *Computer Science*. (2015, November). Grasping the Bytes. Retrieved March 21, 2020, from http://thomas.baudel.name/Informatique/

8. Nelson, P. (1997). The UPIC system as an instrument of learning. Organised Sound, 2(1), 35–42. doi: 10.1017/s1355771897000083

9. Levin, G. (2000). *Painterly Interfaces for Audiovisual Performance*. M.S. Thesis. MIT Media Lab, Cambridge, MA.

10. Thiebaut, J.-B., Healey, P. G., & Kinns, N. B. (2008). Drawing electroacoustic music.

11. Schedel, M. (2018). Colour is the Keyboard: Case Studies in Transcoding Visual to Sonic. In A. McLean & R.T. Dean (Eds.), *The Oxford Handbook of Algorithmic Music* (pp.401-422). New York, NY: Oxford University Press.

12. Giannakis, K. (2006). A comparative evaluation of auditory-visual mappings for sound visualisation. Organised Sound, 11(3), 297–307. doi: 10.1017/s1355771806001531

13. Kreichi, S. (1995). The ANS Synthesizer: Composing on a Photoelectronic Instrument. Leonardo, 28(1), 59. doi: 10.2307/1576159

14. Rodolphe Bourotte. (2018, July 4). UPISketch: The Renewal of an Old Idea [Paper presentation]. 15th Sound and Music Computing Conference (SMC2018): Limassol, Cyprus. doi: 10.5281/1422637

15. Bourotte, R., & Kanach, S. (2019). UPISketch: The UPIC idea and its current applications for initiating new audiences to music. Organised Sound, 24(3), 252–260. doi: 10.1017/s1355771819000323

16. Roads, C. (1996). *The computer music tutorial*. Cambridge, MA: MIT Press.

17. Park, J. (2020, January 17). *NoteSynth_test_1* [Video file]. Retrieved from https://vimeo.com/385523574