Visualizations Guide
MuVis - an Audio Visualizer app for Music

by Keith Bromley, San Diego, CA, USA kbromley@me.com

(updated for MuVis version 1.1 on 22 Dec 2022)

There are currently twenty-seven visualizations included in the MuVis app. They can be divided into three categories:

(1) one-dimensional spectrum graphs,
(2) two-dimensional spectra (i.e., octave-aligned spectra) and
(3) aesthetically-pleasing dynamic depictions of the spectral history.

Most of these visualizations were transcribed into Swift / SwiftUI from versions I had previously written in java for Besmir Beqiri's "Polaris" audio player project (www.logicbind.io/polaris) over the previous several years. I wish to thank Besmir Beqiri for all of his help in our collaboration.

These pictures and descriptions are correct as of December 2022. Since the MuVis app will continue to evolve, and since the documentation might not be updated in synchronization with the code, these may differ slightly from what you see in future versions of the MuVis app.

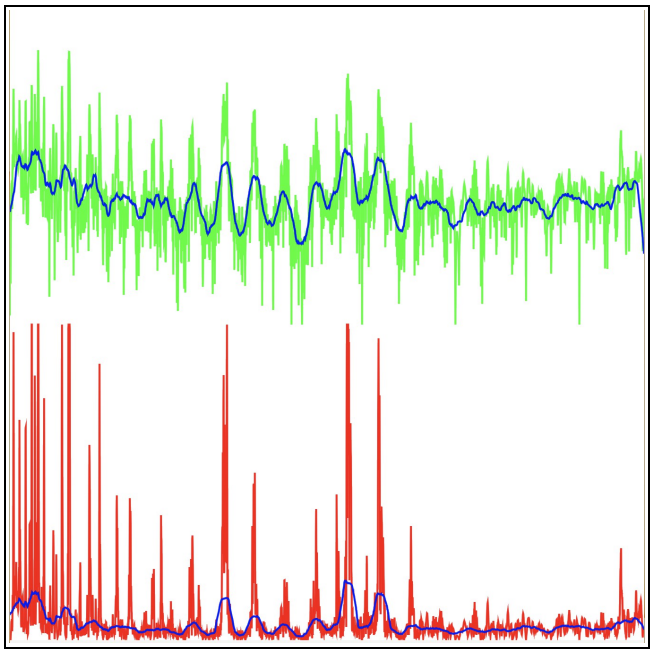
Maybe it's the scientist in me, but I've chosen to write this Visualization Guide as a tutorial introduction to digital signal processing. I recommend that you proceed from start to end because the descriptions of one visualization might serve as a primer to help better understand the following one.

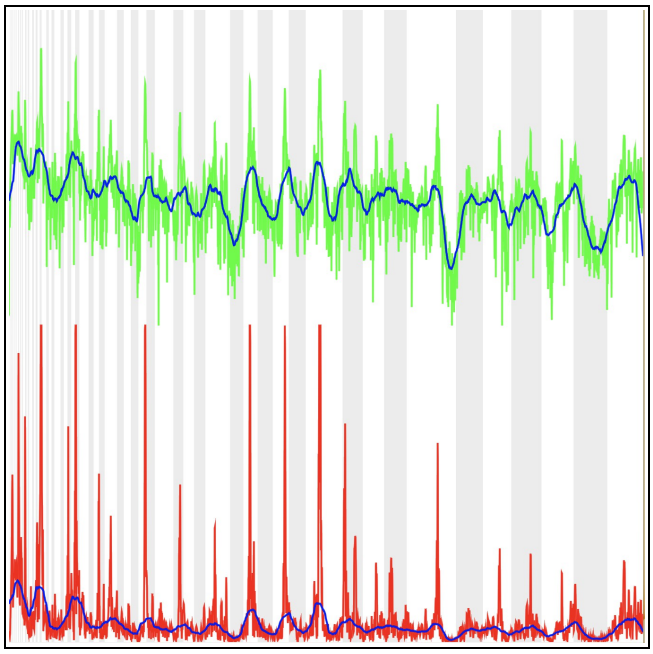
For this document, the visualization pictures I have selected show a white background ("light mode") for the first several, and a black background ("dark mode") for the later ones. Feel free to look at both and choose your favorite.

Also, for this document, I show two pictures for each visualization. The leftmost one is without the Option button being clicked, and the rightmost one is with the Option button clicked. Note that these two pictures sample different audio frames within the selected song track so the curves of the plot will undoubtedly look different.

For each one, here's a brief description of what I attempted to capture in the algorithms, coding, and graphics:

Spectrum





Most of the visualizations in the MuVis app use spectral analysis. In the technical field of "digital signal processing (DSP)", this is performed by an algorithm called the "short-time Fourier transform (STFT)" - which, in turn, uses a very efficient implementation of the Fourier transform called the "FFT". For background knowledge, I suggest you google these terms. (And Wikipedia articles are very helpful.)

In the MuVis app, we use an audio sample rate of 44,100 samples-per-second, and an FFT of length 16,384 samples. This FFT outputs 8,192 spectral values (termed "frequency bins") for each frame of audio samples processed. A very important point to note is that these bins are based upon *linear frequency*. That is, the 8,192 bins stretch from 0 Hz to 22,050 Hz – with a constant additive increment between frequency bins.

This view renders a visualization of the simple one-dimensional spectrum of the music. It renders only the first 755 of the 8,192 bins. The horizontal axis is linear frequency (from 0 Hz on the left to 2,033 Hz on the right).

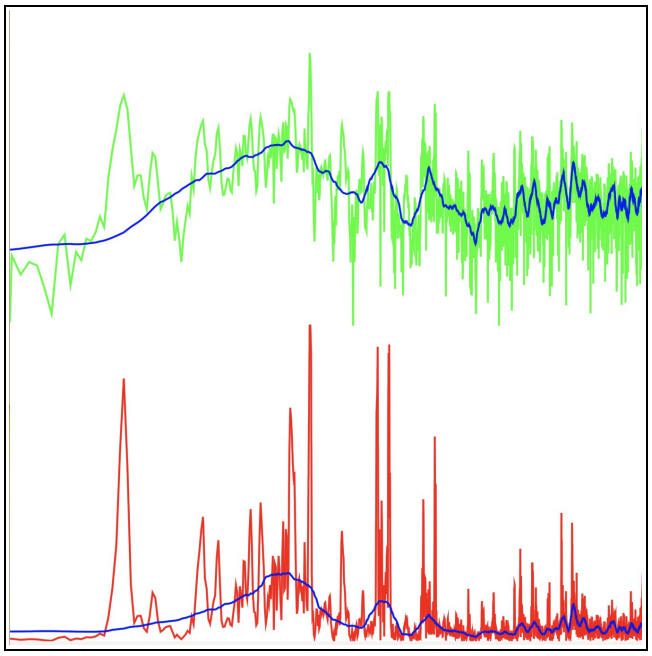
In the lower plot, the vertical axis shows (in red) the mean-square amplitude of the instantaneous spectrum of the audio being played. The red peaks are spectral lines depicting the harmonics of the musical notes. The blue curve is a smoothed average of the red curve (computed by the findMean function within the SpectralEnhancer class). The blue curve typically represents percussive effects which smear spectral energy over a broad range.

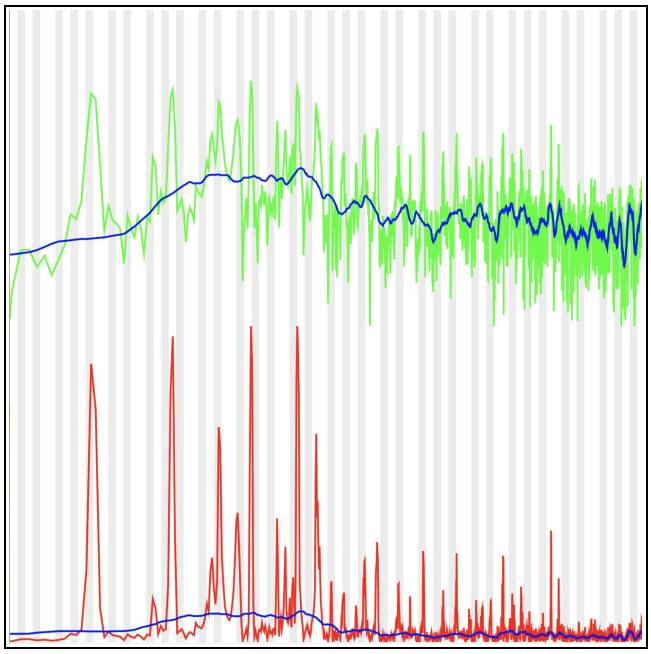
The upper plot (in green) is the same as the lower plot except the vertical scale is decibels (over an 80 dB range) instead of the mean-square amplitude. This more closely represents what the human ear actually hears.

In the right-hand figure above, we have added a piano-keyboard overlay – by clicking on the Option button. Note that the keyboard has been purposely distorted in order to represent musical notes on the linear frequency of the plot. This graphically illustrates the difference between the *linear-frequency* spectrum produced by the FFT and the *logarithmic-frequency* spectrum used in music.

Two characteristics of musical notes are that (1) there are 12 semitones in an octave, and (2) an octave is a frequency doubling. In the right-hand figure above, the entire right-hand half of the spectrum represents just the top octave. (Count the 12 thick white and black keys.) Immediately to the left of this top octave is the next-lower octave (occupying one-quarter of the spectrum). (Count the 12 slightly-thinner white and black keys.) Immediately to the left of this octave is the next-lower octave (occupying one-eighth of the spectrum). (Count the 12 much-thinner white and black keys.)

Music Spectrum





We saw in the Spectrum visualization that the FFT produces linearly-spaced frequency bins (with an additive constant separating them) whereas music uses logarithmically-spaced note frequencies (with a multiplicative constant separating them). (The constant is the twelfth root of two.)

The above figures show what happens if I plot the spectral values for these same 755 frequency bins on a logarithmic horizontal axis. I call this the "Music Spectrum". We immediately see that we are under-sampled at the spectrum's lower frequencies and over-sampled at the spectrum's higher frequencies. This graphically illustrates the crux of the problem of using the FFT for musical analysis. To the best of my knowledge, there is no algorithm providing logarithmically-scaled frequency bins with anywhere near the efficiency and simplicity of the FFT. So my best judgement says to stay with the FFT and just make wise trade-offs.

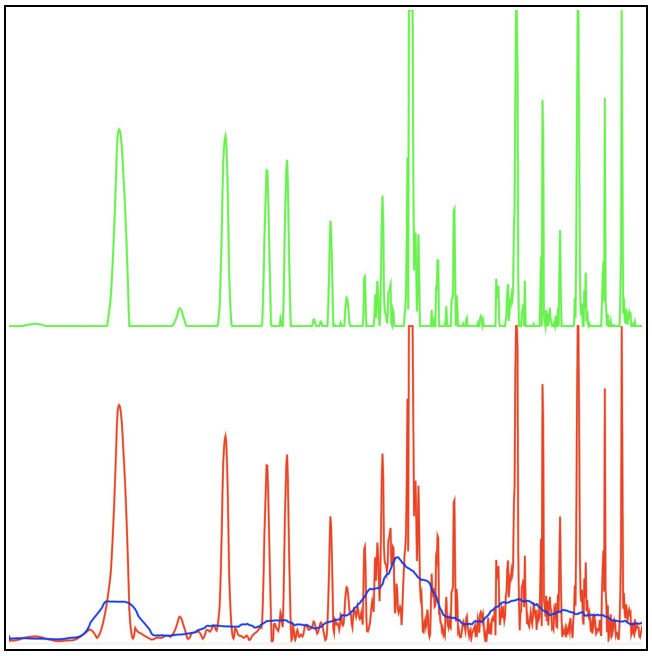
The above plots render the spectrum from frequency bin 12 (corresponding to about 32 Hz) to bin 755 (corresponding to about 2,033 Hz). This range covers the 6 octaves from musical note C1 to musical note B6.

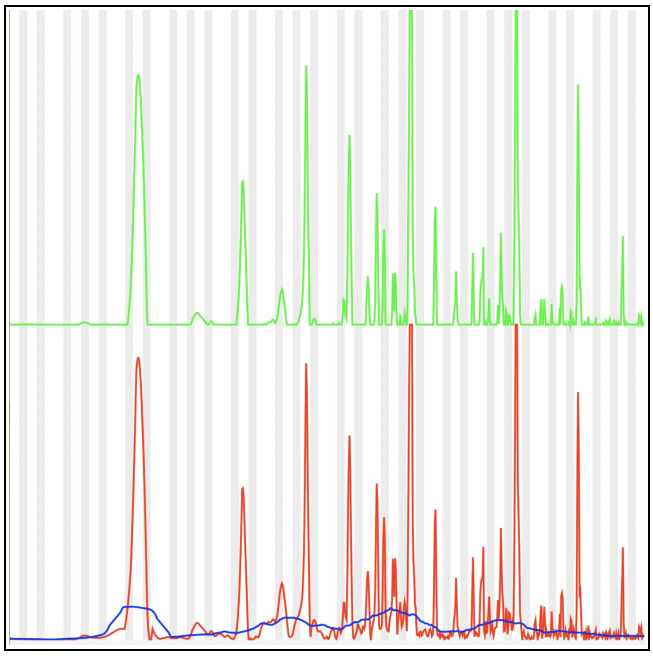
As before, the vertical axis of the lower plot (shown in red) depicts the spectrum's mean-square amplitude and the upper plot (in green) shows the decibel scale over an 80 dB range. Also as before, the blue curves show a smoothed frequency-averaging. In the right-hand figure above, we have added a 72-note keyboard overlay. Note that this time, the keyboard does not have to be distorted (since we are using a logarithmic horizontal axis).

I chose the FFT length (16,384 samples) to provide about 1 bin per note in the bottom octave (32 to 64 Hz). This produces 24 bins in the second octave, 48 bins in the next higher octave, …, up to about 378 bins in the top octave – way more than we need.

A limitation of the Apple AVAudioEngine code that MuVis uses to calculate the FFT of the audio signal is that it computes a new spectrum only every 1/10 of a second. In previous versions of MuVis, this resulted in re-rendering the screen at only 10 frames per second – which looked very "jerky". In MuVis version 1.1, we use the new "Animatable Vector" technique to allow the Apple graphics engine to calculate what the spectrum values would be at 6 time samples within this 1/10 of a second update rate (using linear interpolation). This renders the spectrum at 60 frames per second resulting in a much smoother display. We use this for the Spectrum, Music Spectrum, MuSpectrum, LinearOAS, Overlapped Octaves, Octave Aligned Spectrum, Elliptical OAS, Spiral OAS, Harmonic Alignment (1 and 2), TriOct Music Spectrum, TriOct MuSpectrum, Overlapped Harmonics, Harmonograph (1 and 2), and Superposition visualizations.

MuSpectrum





The first visualization (Spectrum) showed the spectrum on a linear-frequency scale. The second (Music Spectrum) showed the same values on a logarithmic-frequency scale. Now we present a third way of specifying these spectral values – namely passing these values through a quadratic resampling filter. I have coined the name "muSpectrum" for the exponentially-resampled version of the spectrum to more closely represent the notes of the musical scale.

The Spectrum visualization showed these spectral values on a linear-frequency scale. The mathematical problem is to (1) pass this signal through a quadratic smoothing function to interpolate the values between the spectrum samples, and then to (2) resample this signal at points corresponding to the logarithmic-frequency scale of the musical notes. I choose to have 12 points per musical note, and there are 12 notes per octave. Hence we get 144 points per octave. The MuSpectrum visualization (shown above) covers 6 octaves (or 6 x 144 = 864 points). The horizontal axis is linear in terms of points (stretching linearly from 0 on the left to 864 on the right).

The Music Spectrum required the complexity of using a logarithmic horizontal axis, whereas the MuSpectrum allows the convenience of using a linear (in terms of points) horizontal axis.

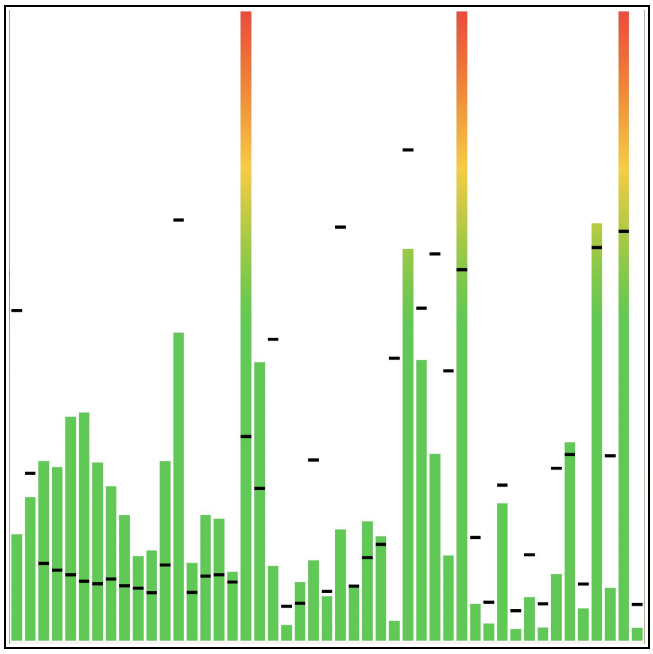
In the lower plot, the horizontal axis is linear (in terms of points) but exponential (in terms of frequency) - from the note C1 (about 33 Hz) on the left to the note B6 (about 1,976 Hz) on the right. The vertical axis shows (in red) the mean-square amplitude of the instantaneous muSpectrum of the audio being played. The red peaks are spectral lines depicting the harmonics of the musical notes being played - and cover six octaves. The blue curve is a smoothed average of the red curve (computed by the findMean function within the SpectralEnhancer class). The blue curve typically represents percussive effects which smear spectral energy over a broad range.

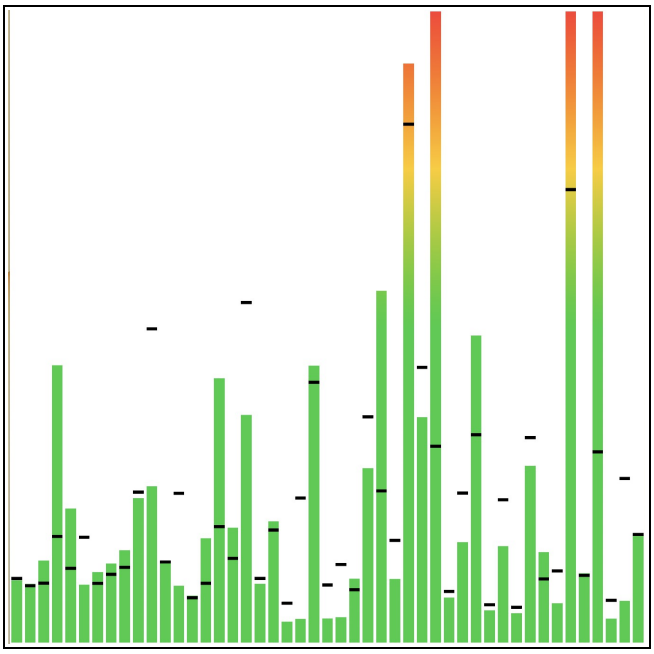
In the upper plot, the green curve is simply the red curve after subtracting the blue curve. This green curve would be a good starting point for analyzing the harmonic structure of an ensemble of notes being played to facilitate automated note detection.

In the right-hand figure above, we have added a 72-note keyboard overlay – by clicking on the Option button. This button also shows the 16 loudest spectral peaks as short black bars at the top of each spectrum.

In the remaining visualizations, I sometimes use the Music Spectrum (when wanting to accurately capture the samples at the higher frequencies) and sometimes use the muSpectrum (when wanting the convenience of using a linear scale of the musical notes). I will strive to make it clear in this guide which I am using.

Spectrum Bars





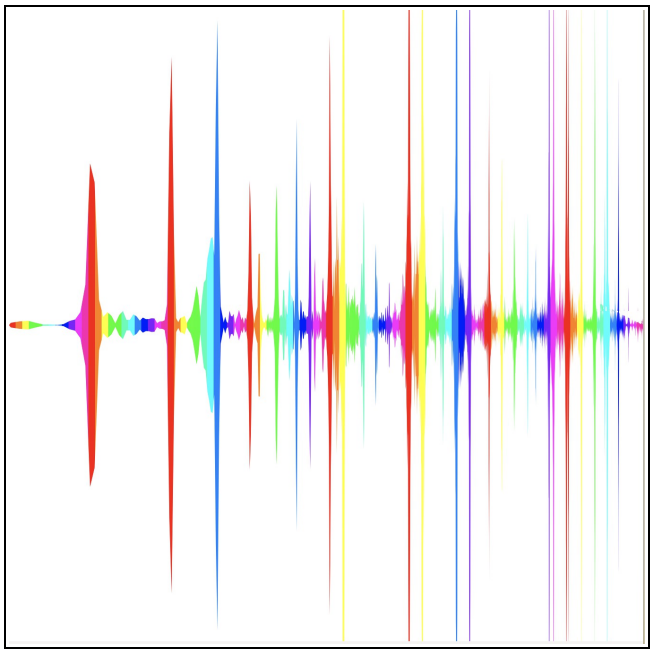
This is basically the same as the muSpectrum plot - but with much fancier dynamic graphics. The horizontal axis is "note number" - covering the 48 notes (four octaves) between C2 and B5.

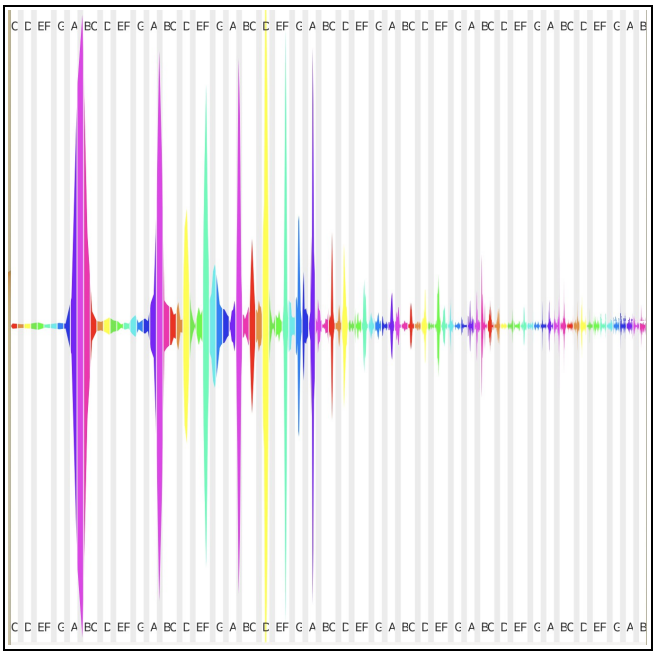
This visualization was adapted from Matt Pfeiffer's tutorial "Audio Visualizer Using AudioKit and SwiftUI". See audiokitpro.com/audiovisualizertutorial . And see github.com/Matt54/SwiftUI-AudioKit-Visualizer for the source code.

I believe that this is copied from the FFTView visualization within the AudioKitUI framework. See github.com/AudioKit/AudioKitUI/tree/main/Sources/AudioKitUI/Visualizations .

When clicking the Option button, the dynamics of the display changes becomes slower. Unfortunately, the right-hand picture above cannot capture the time-varying dynamics of the Visualization.

Linear OAS





The LinearOAS visualization is similar to the Music Spectrum visualization in that it shows an amplitude vs. exponential-frequency spectrum of the audio waveform. The horizontal axis extends from frequency bin 12 (about 32 Hz) to bin 3,021 (about 8,134 Hz). This range covers 96 notes (i.e., 8 octaves) from musical note C1 to musical note B8. For a pleasing effect, the vertical axis shows both an upward-extending Music Spectrum in the upper-half screen and a downward-extending Music Spectrum in the lower-half screen.

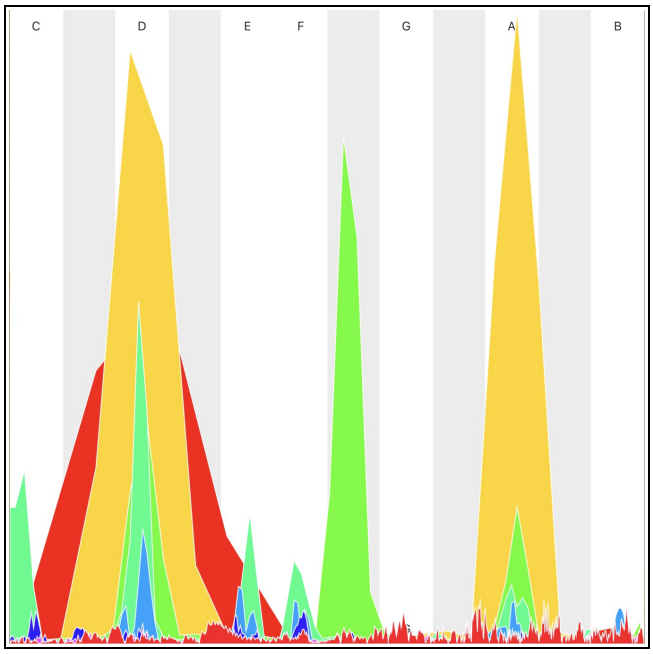
When the Option button is clicked, the visualization adds a piano-keyboard overlay to clearly differentiate the black notes (in gray) from the white notes (in white). Also, it adds note names for the white notes at the top and bottom.

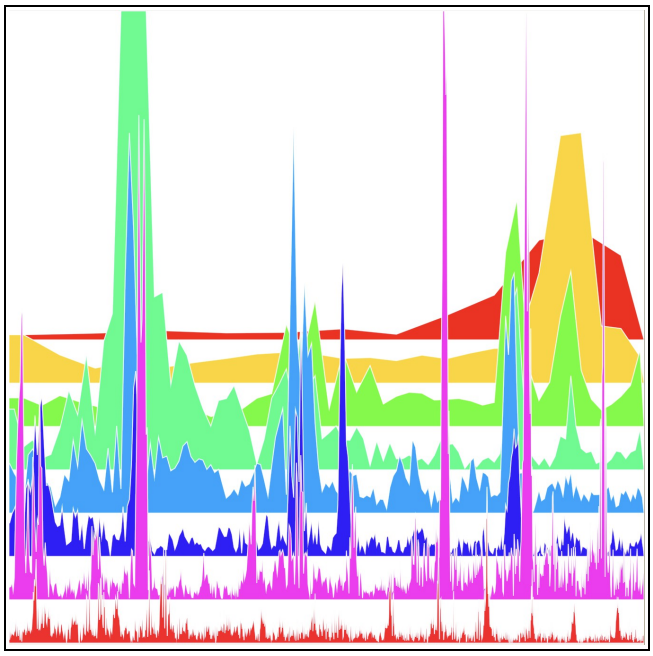
The spectral peaks comprising each note are a separate color. The colors of the grid are consistent across all octaves - hence all octaves of a "C" note are red; all octaves of an "E" note are green, and all octaves of a "G" note are light blue, etc. Many of the subsequent visualizations use this same note coloring scheme. I have subjectively selected these to provide high color difference between adjacent notes.

NoteC_Color	red = 1.0	green = 0.0	blue = 0.0	// red
NoteCsharp_Color	red = 1.0	green = 0.5	blue = 0.0	
NoteD_Color	red = 1.0	green = 1.0	blue = 0.0	// yellow
NoteDsharp_Color	red = 0.1	green = 1.0	blue = 0.0	
NoteE_Color	red = 0.0	green = 1.0	blue = 0.0	// green
NoteF_Color	red = 0.0	green = 1.0	blue = 0.7	
NoteFsharp_Color	red = 0.0	green = 1.0	blue = 1.0	// cyan
NoteG_Color	red = 0.0	green = 0.5	blue = 1.0	
NoteGsharp_Color	red = 0.0	green = 0.0	blue = 1.0	// blue
NoteA_Color	red = 0.5	green = 0.0	blue = 1.0	
NoteAsharp_Color	red = 1.0	green = 0.0	blue = 1.0	// magenta
NoteB_Color	red = 1.0	green = 0.0	blue = 0.7	

The visual appearance of this spectrum is of each note being rendered as a small blob of a different color. However, in fact, the code implements this effect by having static vertical blocks depicting the note colors and then having the non-spectrum rendered as one big white /dark-gray blob covering the non-spectrum portion of the spectrum display. The static colored vertical blocks are rendered first; then the dynamic white / dark-gray big blob; then the gray "black notes"; and finally the note names.

Overlapped Octaves





The OverlappedOctaves visualization is a variation of the upper half of the LinearOAS visualization - except that it stacks notes that are an octave apart. That is, it has a grid of one row tall and 12 columns wide. All of the "C" notes are stacked together (in Swift, a ZStack) in the left-hand box; all of the "C#" notes are stacked together in the second box, etc.

We overlay a stack of 8 octaves of the Music Spectrum with the lowest-frequency octave at the back, and the highest-frequency octave at the front. Octave 0 is rendered in red; octave 1 is magenta; octave 2 is blue, etc. The octave's lowest frequency is at the left pane edge, and it's highest frequency is at the right pane edge.

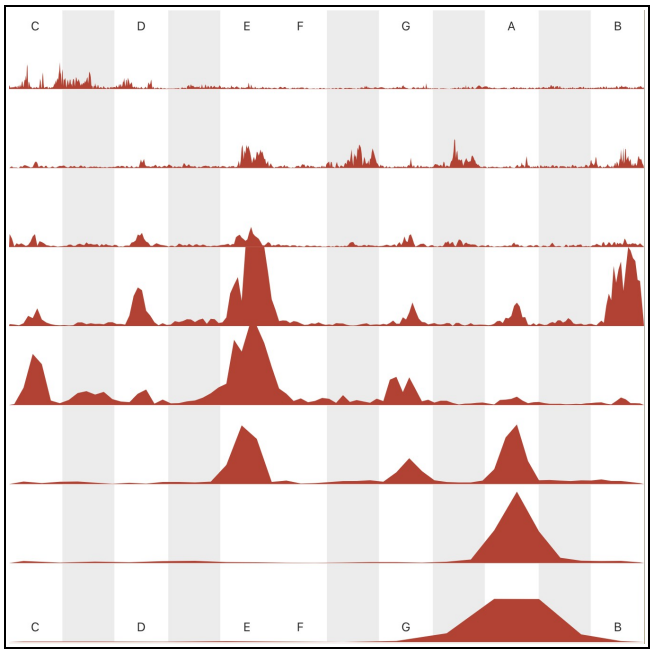
Each octave is a standard spectrum display (converted from linear to exponential frequency) covering one octave. Each octave is overlaid one octave over the next-lower octave. (Note that this requires compressing the frequency range by a factor of two for each octave.)

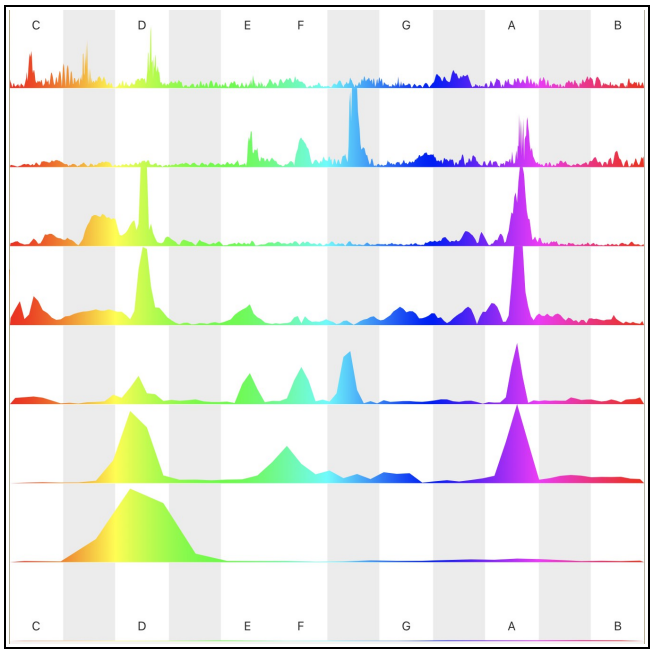
The leftmost column represents all of the musical "C" notes, that is: notes 0, 12, 24, 36, 48, 60, 72, and 84. The rightmost column represents all of the musical "B" notes, that is: notes 11, 23, 35, 47, 59, 71, 83, and 95.

Overlaying this grid is a color scheme representing the white and black keys of a piano keyboard. Also, the name of the note is displayed in each column.

If the Option button is clicked, the rendering of each octave dynamically moves up and down – as shown in the right-hand picture above.

Octave Aligned Spectrum (OAS)





The OctaveAlignedSpectrum (OAS) visualization is one of the bedrock visualizations of this app. (It is my personal favorite.) The basic simplicity of the concept belies a very powerful tool for music visualization.

It is similar to the LinearOAS visualization except that the octaves are laid out one above the other. This is ideal for examining the harmonic structure of the music. (I believe the above left-hand picture shows an A7 chord consisting of the notes A, C#, E, and G. And the above right-hand picture shows a D7 chord consisting of the notes D, F#, A, and C.)

The graphical structure depicted above is a grid of 8 rows by 12 columns. Each of the 8 rows contains all 12 notes within that one octave. Each of the 12 columns contains 8 octaves of that particular note.

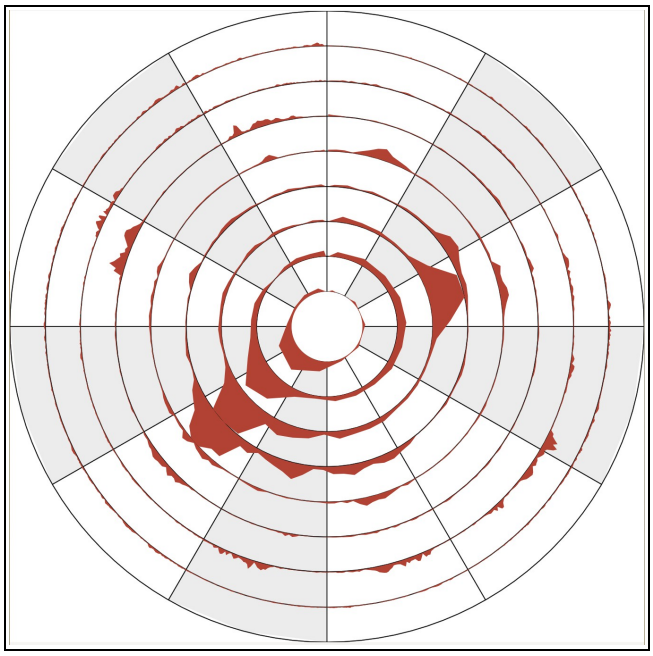
Each octave is a Music Spectrum (i.e., a standard spectrum display converted from linear to exponential frequency) covering one octave. Each octave is overlaid one octave above the next-lower octave. (Note that this requires compressing the frequency range by a factor of two for each octave.)

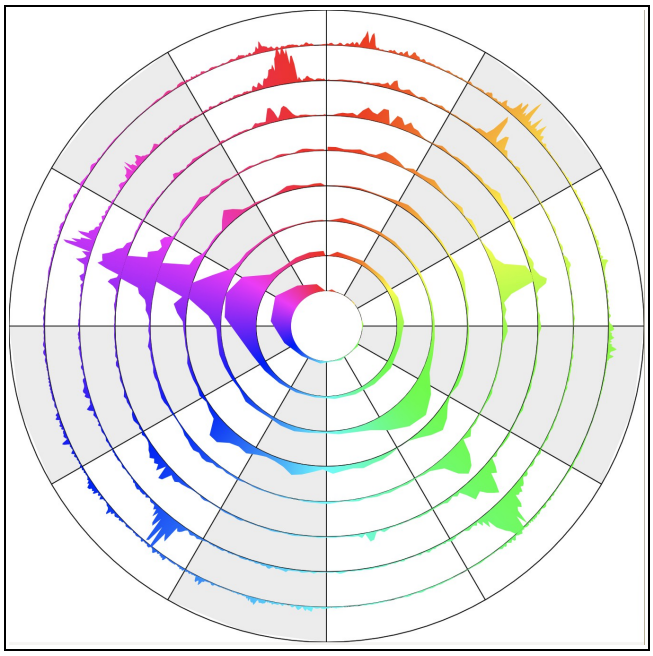
We are using a Music spectrum. The top row shows 1,511 spectral bins (from bin 1,511 to bin 3,021). The next-to-the-top row shows 755 bins (but stretched by a factor of 2 to occupy the same length as the top row). The next-lower-row shows 377 bins (but stretched by a factor of 4 to occupy the same length as the top row). And so on.

Observe that the bottom row contains only 12 bins (from bin 12 to bin 23) whereas the top row contains 1,511bins (12 times two-raised-to-the-eighth-power). The resultant increased resolution at the higher octaves might prove very useful in determining when a vocalist is on- or off-pitch.

In the left-hand picture above, the OAS is a pomegranate red color. When the Option button is clicked, we observe that each one-octave Music Spectrum uses a color gradient corresponding to the standard “hue” range – cycling through red, orange, yellow, green, cyan, blue, magenta, and back to red.

Elliptical OAS





The EllipticalOAS visualization renders the same Music Spectrum information as the OctaveAlignedSpectrum, but displays it as a series of concentric ellipses instead of as rows in a Cartesian grid. Each ellipse of this set is a standard Music Spectrum display covering one octave of frequency. Each ellipse is scaled and aligned one octave above the next-inner ellipse to show the radial alignment of octave-related frequencies in the music.

The parametric equations for a circle are

x = r sin (2 * PI * theta)
y = r cos (2 * PI * theta)

where the angle theta (in radians) is measured clockwise from the vertical axis. Since our rendering pane is rectangular (instead of square), we should generalize this to be an ellipse in order to maximally fill the rendering pane. The parametric equations for an ellipse are

x = A sin (2 * PI * theta)
y = B cos (2 * PI * theta)

where A and B are the major- and minor-radii respectively. (Strictly speaking, theta is no longer the desired geometric angle, but the approximation becomes more accurate as A = B.)

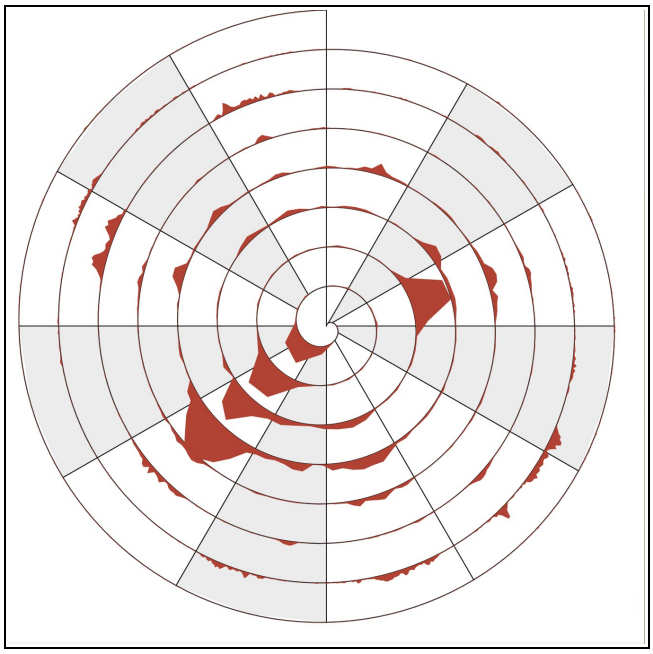
In rendering each of the 8 ellipses (one for each octave of spectral data) we:

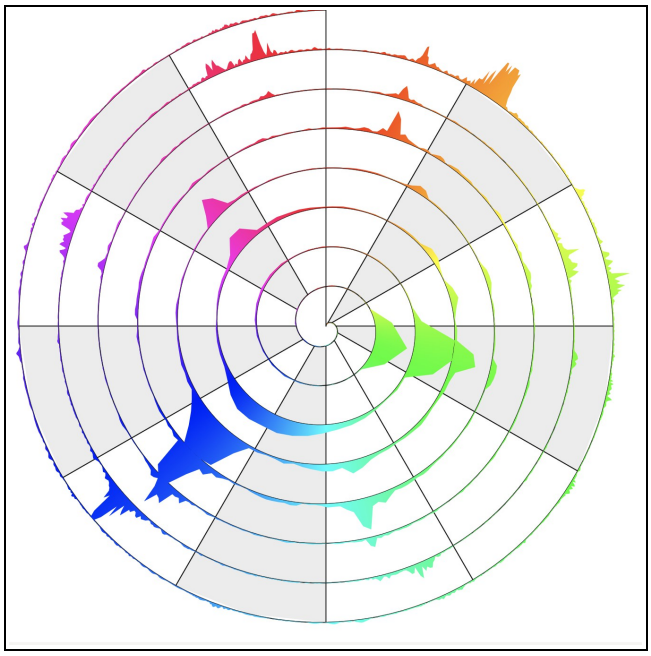
1.) Start defining a multi-segmented line (starting at the twelve o'clock position) on the appropriate ellipse.
2.) Add points to the line as we navigate counter-clockwise around the ellipse.
3.) When we get all the way around to the twelve o'clock position again, we reverse direction and add more points.
4.) During this latter path, we add the spectral bin value to the radius of the ellipse so that the curve has outward bumps representing the spectral peaks.
5.) When we get all the way around to the original starting point, we close the line segments.
6.) Finally, we fill this blob with a desired fill color. (I like the pomegranate red color shown above.)

In the left-hand picture above, the Elliptical OAS spectrum is rendered in pomegranate red. The right-hand picture (obtained when clicking the Option button) uses an angular gradient covering the standard “hue” range - cycling through red, orange, yellow, green, cyan, blue, magenta, and back to red.

The white/gray overlay denotes the white notes and black notes of a piano keyboard.

Spiral OAS





The SpiralOAS visualization renders the same Music Spectrum information as the OctaveAlignedSpectrum, but displays it along a spiral instead of as rows in a Cartesian grid. Each 360-degree revolution of this spiral is a standard Music Spectrum display covering one octave of frequency. Each revolution is scaled and aligned one octave above the next-inner revolution to show the radial alignment of octave-related frequencies in the music.

The LinearOAS visualization had the nice property that all of the Music Spectrum bins were adjacent to their neighbors, but it was poor at identifying harmonic relationships. The OctaveAlignedSpectrum and EllipticalOAS visualizations were much more helpful in showing harmonic spectral relationships and identifying musical notes, but sacrificed the continuity between neighboring spectral bins. (That is, as the audio frequency of a tone increased, it could disappear off of the right end of one row and reappear at the left end of the next-higher row.) This SpiralOAS visualization attempts to get the best of both worlds. It has the same harmonic alignment properties of the EllipticalOAS visualization while having all spectral bins uniformly rendered contiguously along one continuous line (namely, the spiral). In other words, the graphic representation more closely resembles the physics of the sound being analyzed.

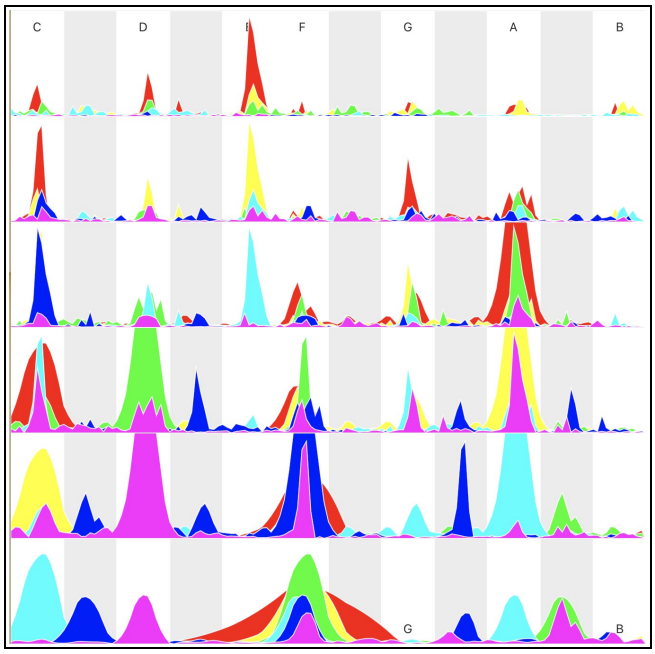
The spiral used in this visualization is called the "Spiral of Archimedes" (also known as the arithmetic spiral) named after the Greek mathematician Archimedes. In polar coordinates, it can be described by the parametric equations

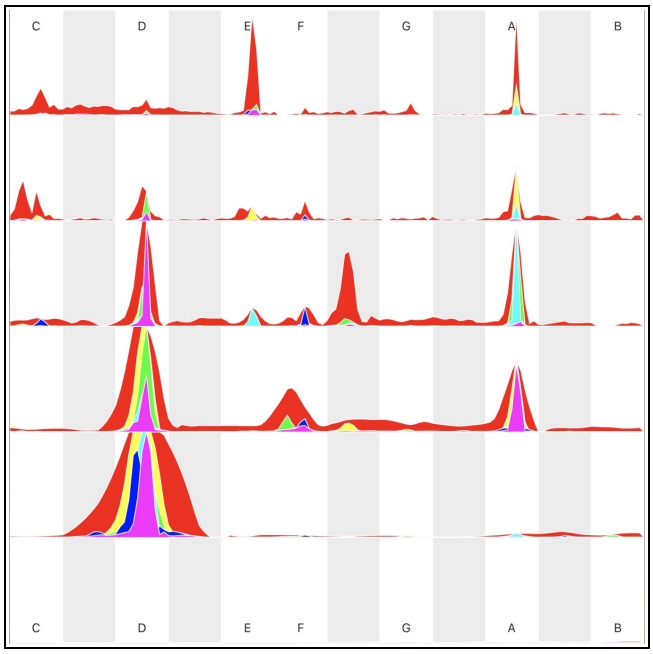
$$x = A * \theta * \sin(2 * PI * \theta)$$
$$y = B * \theta * \cos(2 * PI * \theta)$$

where theta is the angle in radians (measured clockwise from the 12 o'clock position).

A Google search uncovered the patent US5127056A for a "Spiral Audio Spectrum Display System". See www.google.com/patents/US5127056 . It was filed in 1990 by Allen Storaasli. It states that "Each octave span of the audio signal is displayed as a revolution of the spiral such that tones of different octaves are aligned and harmonic relationships between predominant tones are graphically illustrated."

Harmonic Alignment





The HarmonicAlignment visualization is an enhanced version of the OctaveAlignedSpectrum visualization. It renders a muSpectrum displaying the FFT frequency bins (converted to "points" by a resampling filter) of the live audio data on a two-dimensional Cartesian grid. Each row of this grid is a standard muSpectrum display covering one octave. Each row is aligned one octave above the next-lower row to show the vertical alignment of octave-related frequencies in the music. Hence the six rows of our displayed grid cover six octaves of musical frequency.

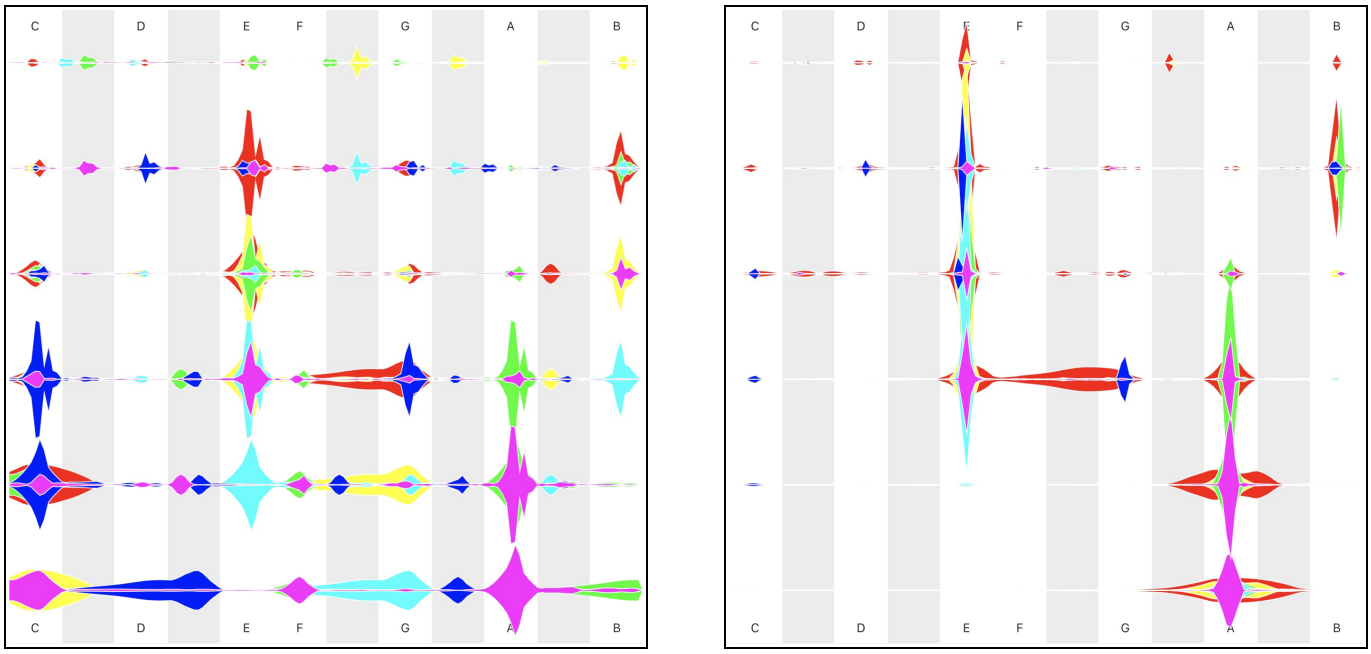
The bottom 6 octaves (the 72 notes from 0 to 71) will be displayed as possible fundamentals of the notes in the music, and the remaining notes (those above 71) will be used only for harmonic information. The novel feature of the HarmonicAlignment visualization is the rendering of the harmonics beneath each note. We are rendering 6 harmonics of each note. We use the following values of hue to assign colors to these harmonics:

hue = 0	red	harm = 1 (fundamental)
hue = 1/6	yellow	harm = 2
hue = 2/6	green	harm = 3
hue = 3/6	cyan	harm = 4
hue = 4/6	blue	harm = 5
hue = 5/6	magenta	harm = 6

The fundamental (harm=1) (the basic octave-aligned spectrum of the fundamentals) is shown in red. The first harmonic (harm=2) shows as yellow; the second harmonic (harm=3) is green; and the third harmonic (harm=4) is cyan, and so on. It is instructive to note the massive redundancy displayed here. A fundamental note rendered as red in row 4 will also appear as yellow in row 3 since it is the first harmonic of the note one-octave lower, and as green in row 2 since it is the second harmonic of the note two-octaves lower, and as cyan in row 1 since it is the fourth harmonic of the note three-octaves lower. This redundancy provides a very vibrant colorful display.

In order to decrease the visual clutter (and to be more musically meaningful), we multiply the value of the harmonics by the value of the fundamental. So, if there is no meaningful amplitude for the fundamental, then its harmonics are not shown (or at least only with low amplitude). This provides a more musically realistic depiction of the muSpectrum (and its harmonics). The right-hand picture above shows this case – obtained by clicking the Option button.

Harmonic Alignment 2

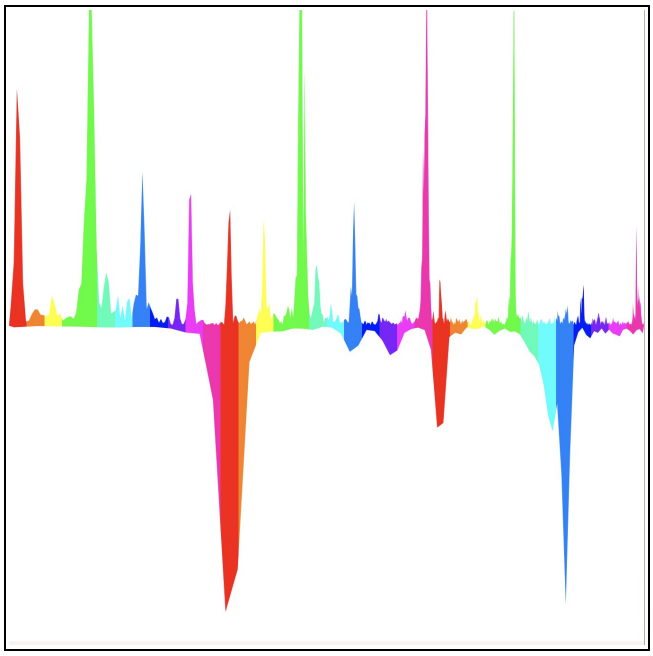


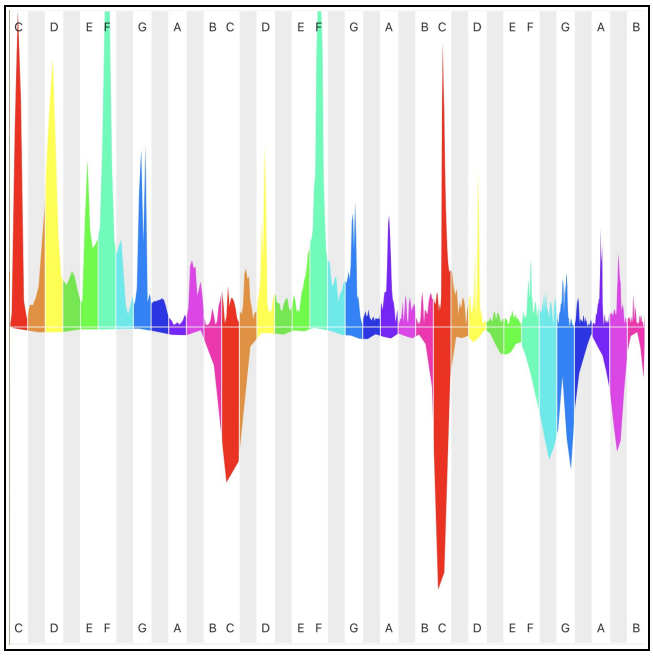
The Harmonic Alignment 2 visualization depicts the same information as the HarmonicAlignment visualization but rendered in a slightly different form. (This is purely for aesthetic effect - which you may find pleasing or annoying.) The muSpectrum for each of the six octaves (and for each of the six harmonics within each octave) is rendered twice - one upward stretching muSpectrum and one downward stretching muSpectrum.

The OAS of the fundamental notes (in red) is rendered first. Then the OAS of the first harmonic notes (in yellow) are rendered over it. Then the OAS of the second harmonic notes (in green) are rendered over it, and so on - until all 6 harmonics are depicted.

Again, in the right-hand picture (obtained by clicking the Option button), we multiply the value of the harmonics by the value of the fundamental. So, the harmonics are shown if-and-only-if there is meaningful energy in the fundamental.

TriOct Music Spectrum





The TriOct Music Spectrum visualization is similar to the LinearOAS visualization in that it shows a Music Spectrum of six octaves of the audio waveform - however it renders it as two separate Music Spectrum displays.

It has the format of a downward-facing Music Spectrum in the lower half-screen covering the lower three octaves, and an upward-facing Music Spectrum in the upper half-screen covering the upper three octaves. Each half screen shows three octaves. (The name "bi-tri-octave Spectrum" seemed unduly cumbersome, so I abbreviated it to "tri-octave spectrum"). The specific note frequencies are:

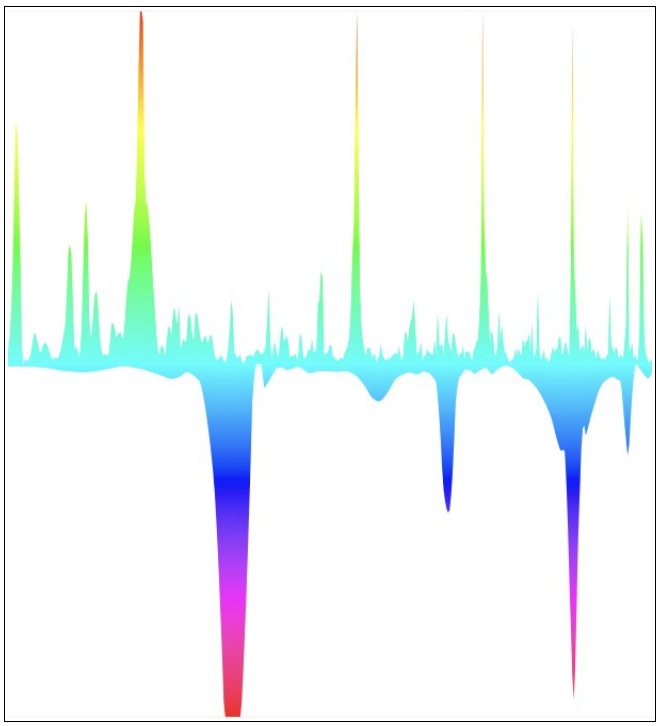
```
262 Hz                  523 Hz                  1046 Hz                 1976 Hz
 C4                      C5                      C6                      B6
 |                       |                       |                       |
 W B W B W W B W B W B W W B W B W W B W B W B W W B W B W W B W B W B W <- white or black keys

 W B W B W W B W B W B W W B W B W W B W B W B W W B W B W W B W B W B W <- white or black keys
 |                       |                       |                       |
 C1                      C2                      C3                      B3
33Hz                    65 Hz                   130 Hz                  247 Hz
```

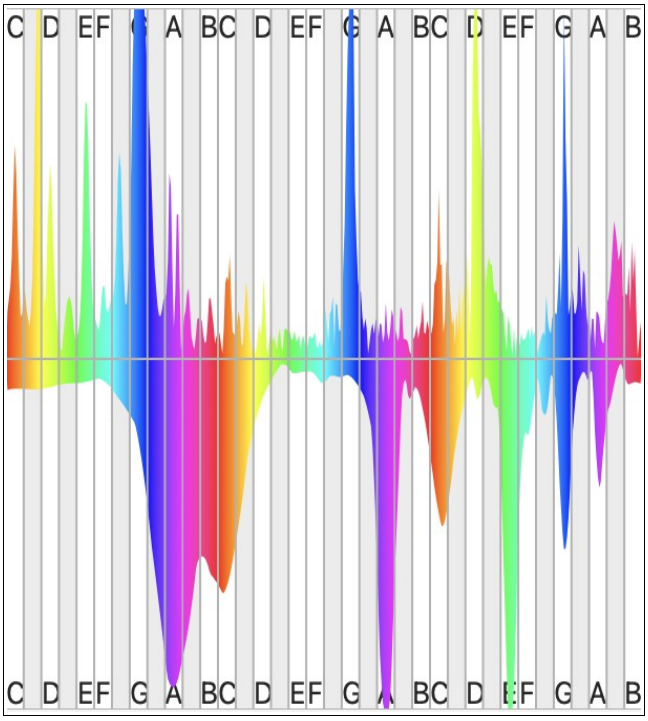
As with the LinearOAS visualization, the spectral peaks comprising each note are a separate color, and the colors of the grid are consistent across all octaves - hence all octaves of a "C" note are red; all octaves of an "E" note are green, and all octaves of a "G" note are light blue, etc.

In the right-hand picture above (obtained by clicking the Option button), we have added a piano-keyboard overlay to clearly differentiate the black notes (in gray) from the white notes (in white). Also, we have added note names for the white notes at the top and bottom.

The visual appearance of these two Music Spectra is of each note being rendered as a small blob of a different color. However, in fact, we implement this effect by having static vertical blocks depicting the note colors and then having the non-spectrum rendered as two big white / dark-gray blobs covering the non-spectrum portion of the spectrum display - one each for the upper-half-screen and the lower-half-screen. The static colored vertical blocks are rendered first; then the dynamic white / dark-gray big blobs; then the gray "black notes"; and finally the note names.

TriOct MuSpectrum





The TriOct MuSpectrum visualization is similar to the LinearOAS visualization in that it shows a muSpectrum of six octaves of the audio waveform - however it renders it as two separate muSpectrum displays. It has the format of a downward-facing muSpectrum in the lower half-screen covering the lower three octaves, and an upward-facing muSpectrum in the upper half-screen covering the upper three octaves. Each half screen shows three octaves. (The name "bi-tri-octave muSpectrum" seemed unduly cumbersome, so I abbreviated it to "tri-octave muSpectrum"). The specific note frequencies are:

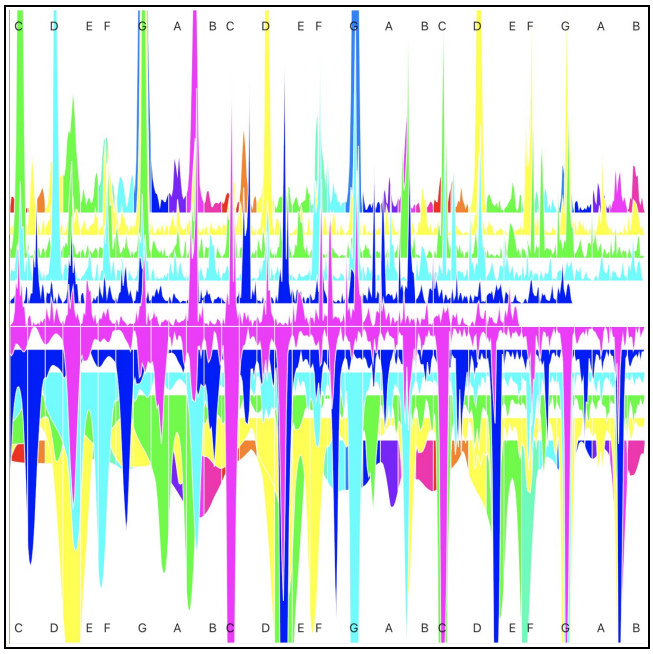
```
262 Hz                  523 Hz                  1046 Hz                 1976 Hz
 C4                      C5                      C6                      B6
 |                       |                       |                       |
 W B W B W W B W B W B W W B W B W B W W B W B W W B W B W B W  <- white or black keys

 W B W B W W B W B W B W W B W B W B W W B W B W W B W B W B W  <- white or black keys
 |                       |                       |                       |
 C1                      C2                      C3                      B3
33Hz                    65 Hz                   130 Hz                  247 Hz
```

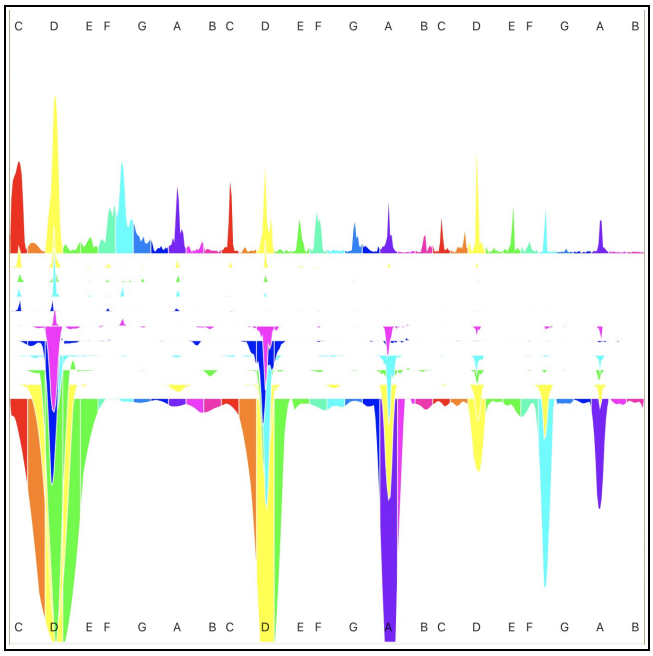
In the left-hand picture above, a vertical color-gradient is applied to the TriOct MuSpectrum. It is the colors of the standard hue (ranging from red at the top through yellow, green, cyan, blue, magenta, and finally to red at the bottom).

In the right-hand picture above (obtained by clicking the Option button), a horizontal color gradient is applied to each of the upper and lower TriOct MuSpectra. The gradient is the color of the standard hue but repeating over each of the three octaves. Also, we have added a piano-keyboard overlay to clearly differentiate the black notes (in gray) from the white notes (in white). Also, we have added note names for the white notes at the top and bottom.

By comparing the pictures of the TriOct Music Spectrum and the TriOct muSpectrum, your can clearly see the differences that we discussed earlier. At the low frequencies, the TriOct Music Spectrum is a series of straight lines connecting the FFT-supplied spectral data points – whereas the TriOct muSpectrum is much smoother. At the high frequencies, the TriOct Music Spectrum has very dense spectral data points (readily emphasizing tall spectral peaks) – whereas the TriOct muSpectrum is much smoother (and loses many of the spectral peaks).

Overlapped Harmonics





The OverlappedHarmonics visualization is similar to the OverlappedOctaves visualization - except, instead of rendering just one octave in each of six rows, it renders 3 octaves in each of two rows. Both show a six-octave muSpectrum. Immediately in front of this "fundamental" muSpectrum is the muSpectrum of the first harmonic (which is one octave above the fundamental spectrum) in a different color. Immediately in front of this first-harmonic muSpectrum is the muSpectrum of the second harmonic (which is 19 notes higher than the fundamental muSpectrum) in a different color. Immediately in front of this second-harmonic is the muSpectrum of the third-harmonic (which is 24 notes above the fundamental.) And so on.

The on-screen display format has a downward-facing spectrum in the lower half-screen covering the lower three octaves, and an upward-facing spectrum in the upper half-screen covering the upper three octaves. The specific frequencies are the same as those of the TriOct muSpectrum.

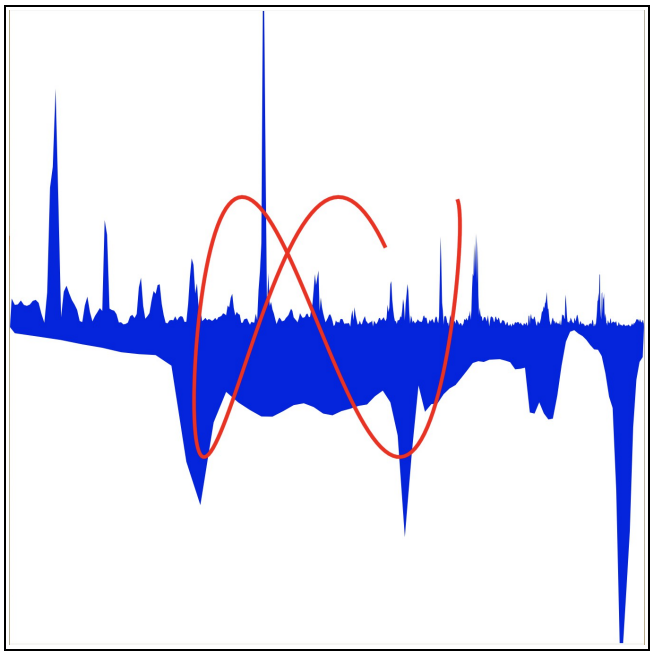
For the fundamental muSpectrum, the spectral peaks comprising each note are a separate color, and the colors of the grid are consistent across all octaves - hence all octaves of a "C" note are red; all octaves of an "E" note are green, and all octaves of a "G" note are light blue, etc.

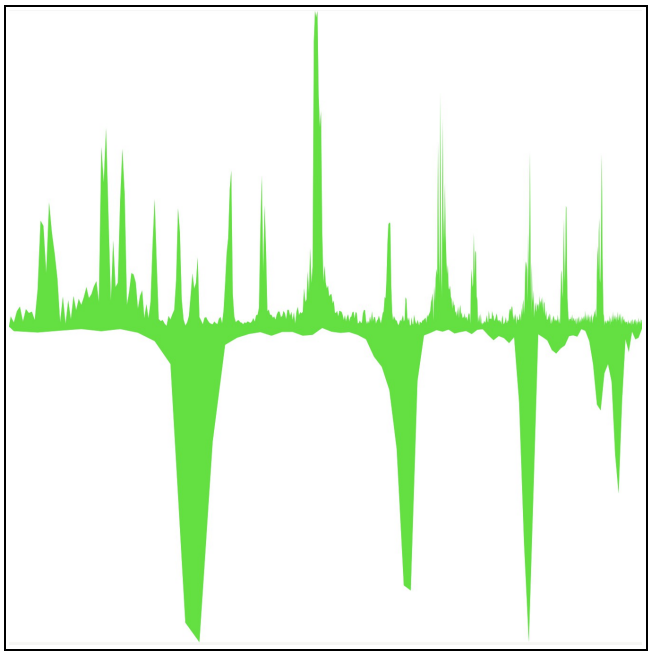
For the harmonics, we show a separate color for each harmonic. The first harmonic (harm=2) shows as yellow; the second harmonic (harm=3) is green; the third harmonic (harm=4) is light blue, the fourth harmonic (harm=5) is dark blue, and the fifth harmonic (harm=6) is magenta.

We have added a keyboard overlay and note names for the white notes at the top and bottom.

In order to decrease the visual clutter (and to be more musically meaningful), we multiply the value of the harmonics by the value of the fundamental. So, if there is no meaningful amplitude for the fundamental, then its harmonics are not shown (or at least only with low amplitude). This provides a more musically realistic depiction of the muSpectrum (and its harmonics). The right-hand picture above shows this case – obtained by clicking the Option button.

Harmonograph





I have long sought to develop a music-visualization scheme that readily displays the harmonic relationship of the frequencies being played. My inspiration comes from Lissajous figures generated by applying sinusoidal waveforms to the vertical and horizontal inputs of an oscilloscope. Inputs of the same frequency generate elliptical curves (including circles and lines). Inputs of different frequencies, where one is an integer multiple of the other, generate "twisted" ellipses. A frequency ratio of 3:1 produces a "twisted ellipse" with 3 major lobes. A frequency ratio of 5:4 produces a curve with 5 horizontal lobes and 4 vertical lobes. Such audio visualizations are both aesthetically pleasing and highly informative.

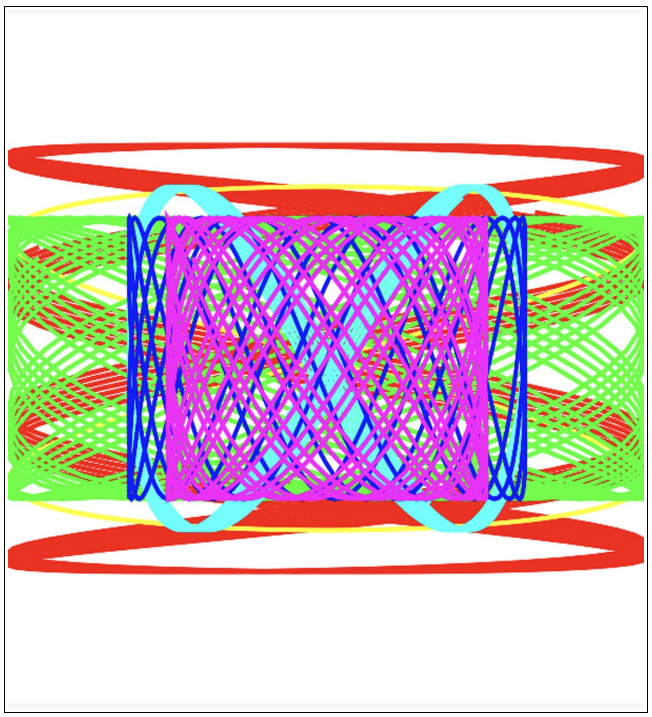
Over the past several years, I have implemented many many such visualizations and applied them to analyzing music. Unfortunately, most suffered from being overly complex, overly dynamic, and uninformative. In my humble opinion, this Harmonograph visualization strikes the right balance between simplicity (i.e., the ability to appreciate the symmetry of harmonic relationships) and dynamics that respond promptly to the music.

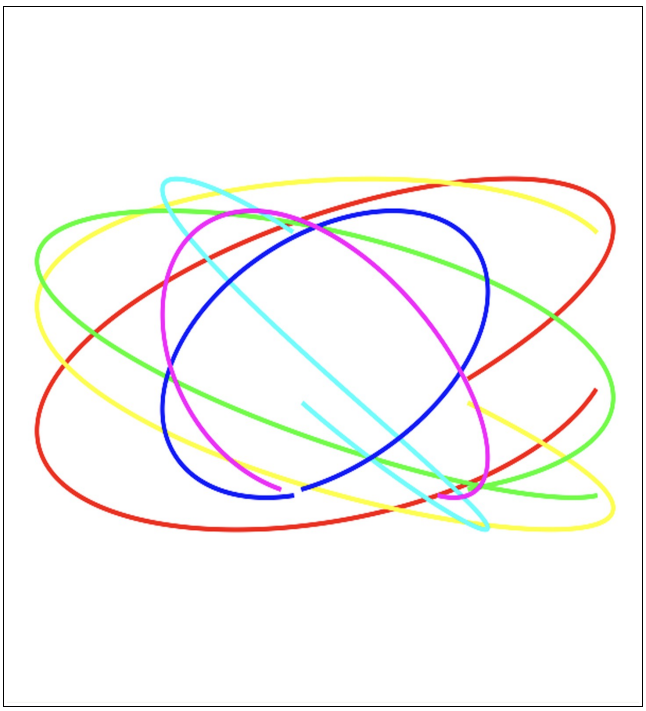
The wikipedia article at https://en.wikipedia.org/wiki/Harmonograph describes a double pendulum apparatus, called a Harmonograph, that creates Lissajous figures from mixing two sinusoidal waves of different frequencies and phases. This MuVis Harmonograph visualization uses just the two loudest spectrum peaks (in the lower three octaves) to produce the Lissajous figure. That is, the loudest peak (in the lower three octaves) generates a sine wave of its frequency to drive the horizontal axis of our visual oscilloscope, and the second-loudest peak (in the lower three octaves) generates a sine wave of its frequency to drive the vertical axis.

If the Option button is clicked, the visualization uses the loudest peak and second-loudest peak from the upper three octaves (instead of the lower three octaves).

For a pleasing effect, the Harmonograph Lissajous figure is rendered on top of a simplified TriOct Music Spectrum visualization.

Harmonograph 2





The Harmonograph 2 visualization has the same motivation as the Harmonograph – except that the horizontal and vertical driving signals for the Lissajous figures come from the loudest four spectral peaks of the muSpectrum.

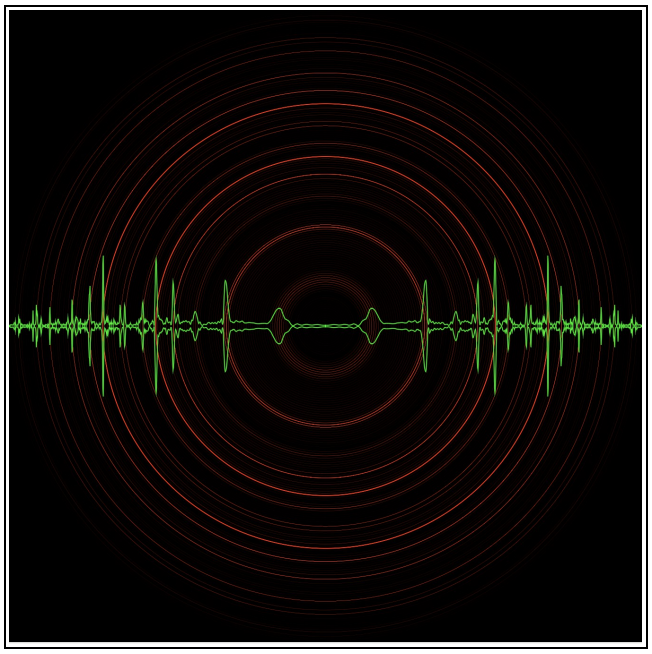
For the given frame of the muSpectrum, let's denote the loudest spectral peak as peak0, the next-loudest peak as peak1, the next-loudest peak as peak2, and the quietest of the four peaks as peak3.

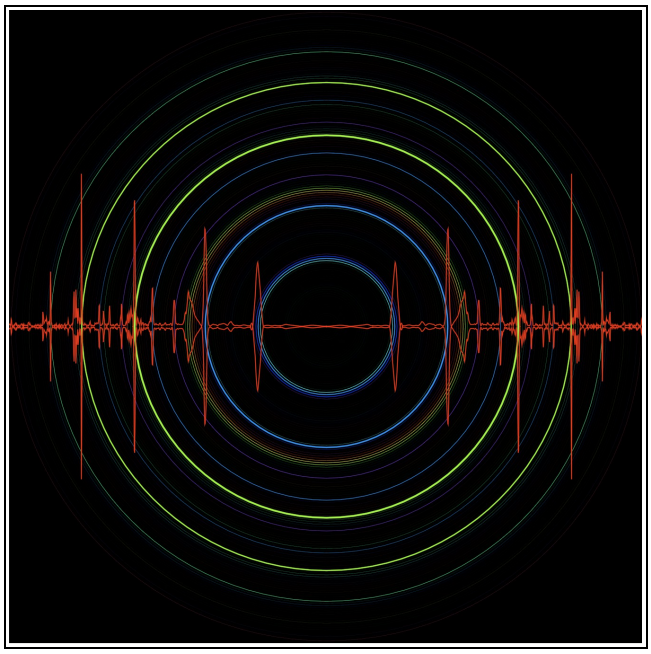
Then the Lissajous figure in red uses the frequencies of peak0 and peak1 as the drivers of it's horizontal and vertical axes. The Lissajous figure in yellow uses peak0 and peak2. That in green uses peak0 and peak3. That in cyan uses peak1 and peak2. That in blue uses peak1 and peak3. And finally, that in magenta uses peak2 and peak3.

If the Option button is clicked, then the frequency of each axis of the Lissajous figure is normalized by the lowest frequency of the octave that the frequency occurs in.

As a reminder, the two figures above were taken with different frames of audio data, so you shouldn't try to equate the peak frequencies in one with the other.

Cymbal





The Cymbal visualization is a different way of depicting the current muSpectrum. It was inspired by contemplating the vibrational patterns of a cymbal. It is purely an aesthetic depiction (with no attempt at real-world modeling).

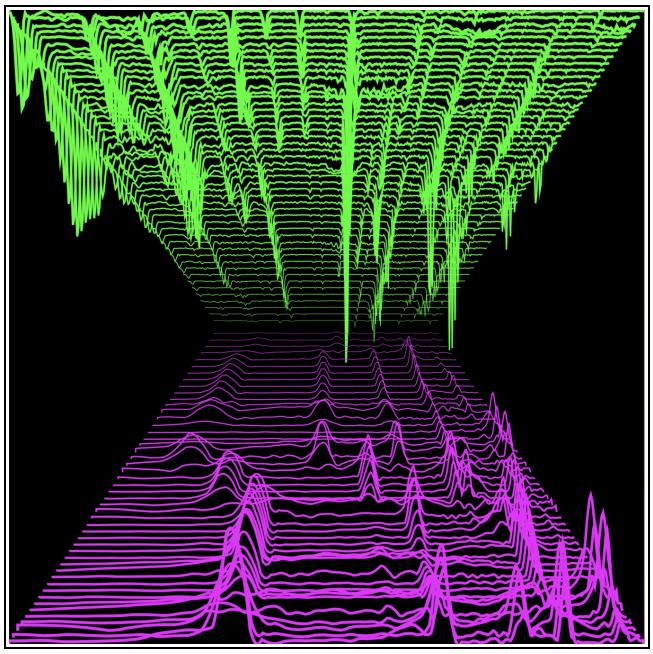
We render 6 octaves of the muSpectrum at 12 notes/octave and 2 points/note. Thus, each muSpectrum contains 6 * 12 * 2 = 144 points. This Cymbal visualization renders 144 concentric circles (all with their origin at the pane center) with their radius proportional to these 144 musical-frequency points. 72 of these are note centers, and 72 are the interspersed inter-note midpoints. We dynamically change the line width of these circles to denote the muSpectrum amplitude.

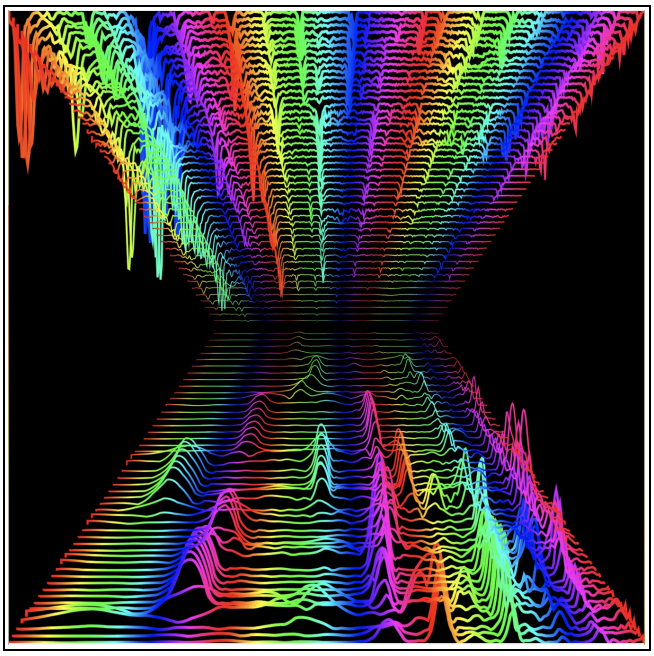
In the normal Cymbal visualization (show in the left-hand picture above), the concentric circles are rendered in red – and may be clipped by the borders of the pane as the width and height of the window are adjusted. And, for aesthetic effect, we overlay a green plot of the current muSpectrum (replicated from mid-screen to the right edge and from mid-screen to the left edge) on top of the circles.

If the Option button is clicked, the red circles become multi-colored ellipses (wherein all of the shapes remain within the visualization pane as the window's width and height are adjusted). Also, the color of the muSpectrum changes from green to red.

As I cycle through the various MuVis visualizations, this is the point at which I typically switch from Light Mode (with a white background) to Dark Mode (with a dark background). This is just a personal preference. I think the remaining visualizations look better with a black background.

Rainbow Spectrum





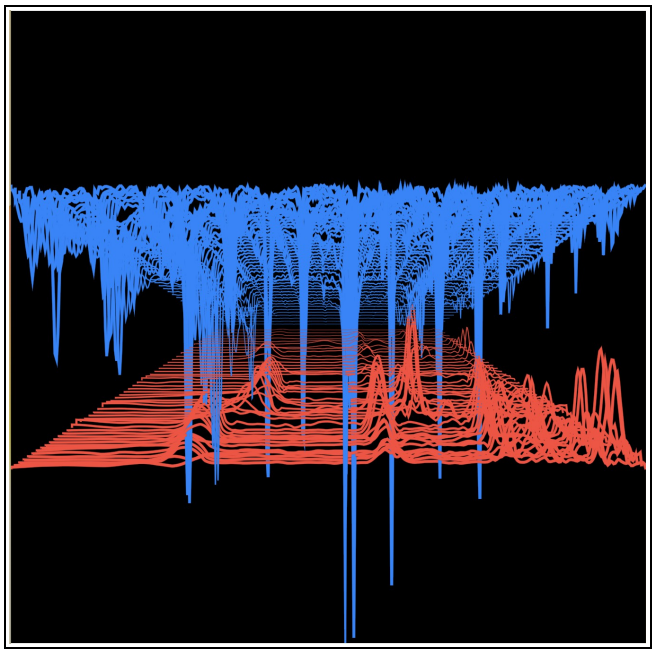
We now start to discuss the MuVis visualizations that depict the time-history of the muSpectra. That is, instead of rendering just the current muSpectrum, they also render the most-recent 48 muSpectra - so they show how the envelope of each note varies with time. With a frame-rate of 0.1 seconds per frame, these 48 muSpectra cover the last 4.8 seconds of the music we are hearing.

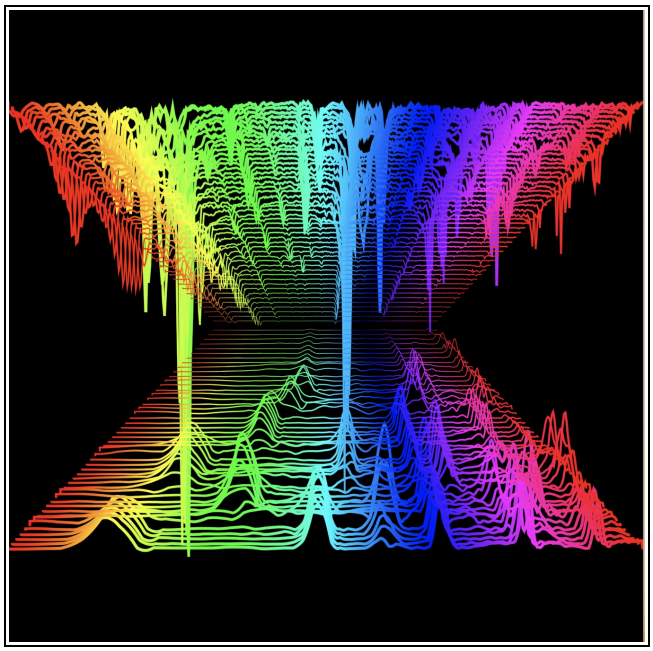
The RainbowSpectrum visualization uses a similar geometry to the TriOctSpectrum visualization wherein the lower three octaves of muSpectrum audio information are rendered in the lower half-screen and the upper three octaves are rendered in the upper half-screen. The current muSpectrum is shown in the bottom and top rows. And the muSpectrum history is shown as drifting (and shrinking) to the vertical mid-screen.

For the normal Rainbow Spectrum visualization, the colors of the upper half-screen and lower half-screen change over time.

If the Option button is clicked (as shown on the right), we observe that each one-octave muSpectrum uses a color gradient corresponding to the standard “hue” range – cycling through red, orange, yellow, green, cyan, blue, magenta, and back to red.

Rainbow Spectrum 2



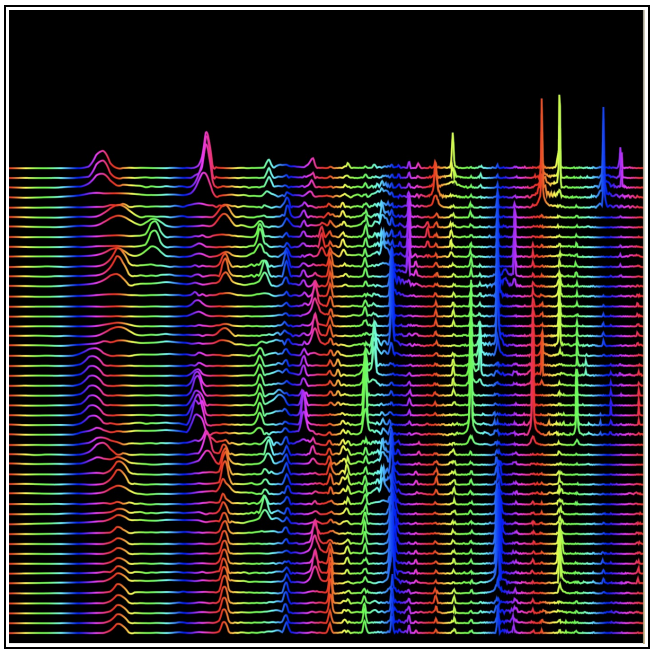


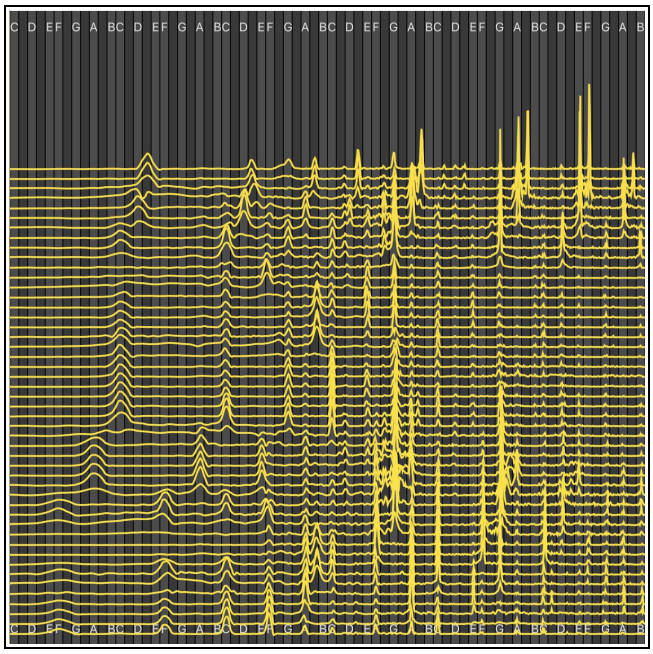
The RainbowSpectrum2 visualization is simply a more dynamic version of the RainbowSpectrum visualization. The rows showing the current muSpectrum are no longer static at the top and bottom of the screen - but move dynamically between the midpoint and the top and bottom of the screen.

For the normal RainbowSpectrum2 visualization, the lower three octaves are colored in red, and the upper three octaves are colored in blue.

If the Option button is clicked (as shown on the right), we observe that each three-octave muSpectrum uses a color gradient corresponding to the standard “hue” range – cycling through red, orange, yellow, green, cyan, blue, magenta, and back to red.

Waterfall





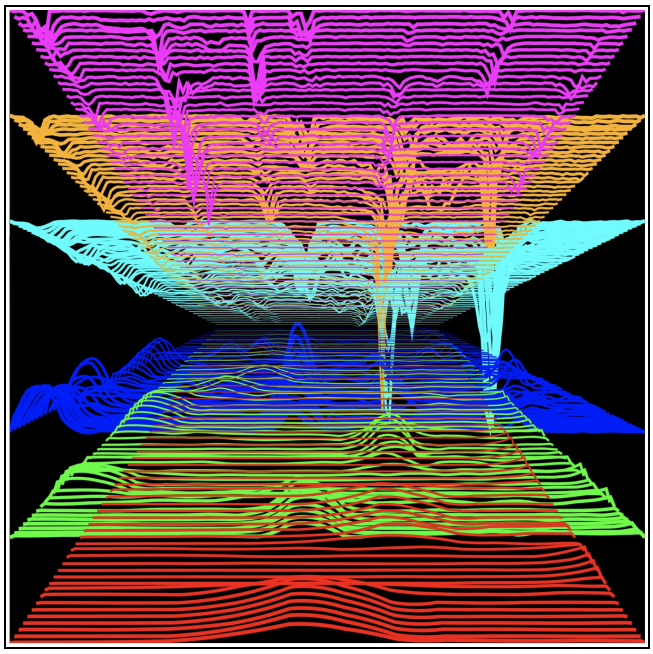
The Waterfall visualization (named because the dynamics resemble a waterfall as the rows move from top to bottom) is a classical old-standby spectral display format used to bring out harmonic structure.

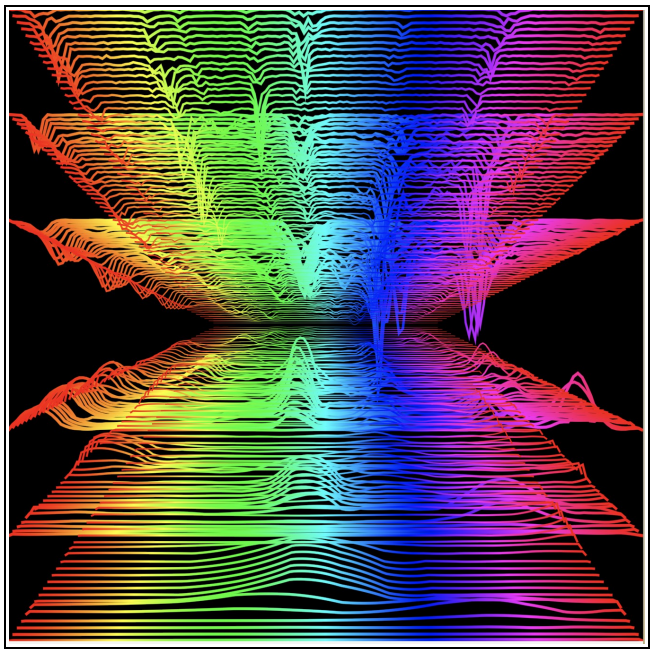
The top row shows the current (most recent) six-octave muSpectrum. The second row shows the six-octave muSpectrum from 0.1 seconds ago. The third row shows the six-octave muSpectrum from 2 * 0.1 = 0.2 seconds ago, and so on. So the overall effect is of the data moving steadily downward – and off the bottom of the pane.

The normal Waterfall visualization has a black background, and each octave of the muSpectrum uses a color gradient corresponding to the standard "hue" range – cycling through red, orange, yellow, green, cyan, blue, magenta, and back to red.

If the Option button is clicked then (as shown on the right above), all rows of the muSpectrum history are the same color (although dynamically changing with time), a six-octave keyboard overlay is added, and the note names for the white notes are shown.

Rainbow OAS





The RainbowOAS visualization uses the same Cartesian grid geometry as the OctaveAlignedSpectrum visualization. However, instead of rendering just the current muSpectrum, it also renders the most-recent 48 muSpectra history - so it shows how the envelope of each note varies with time. Iterative scaling is used to make the older spectral values appear to drift into the background.

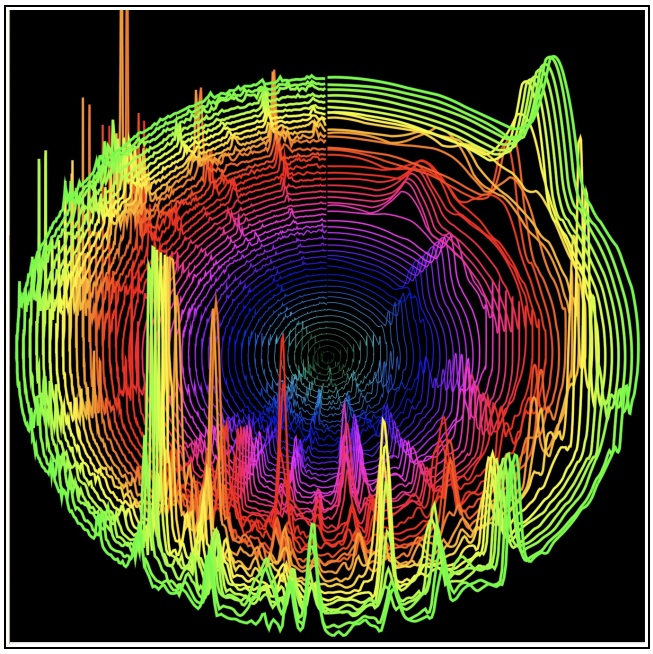
The 6 rows depicted in the above picture cover 6 octaves. Octave-wide spectra are rendered on rows 0, 1, 2 and on rows 4, 5, 6. All iterate towards the vertical-midpoint of the screen. Octave 0 is rendered along row 0 at the bottom of the visualization pane. Octave 5 is rendered along row 6 at the top of the visualization pane. Using a resolution of 12 points per note, each row consists of 12 * 12 = 144 points covering 1 octave. The 6 rows show a total of 6 * 144 = 864 points.

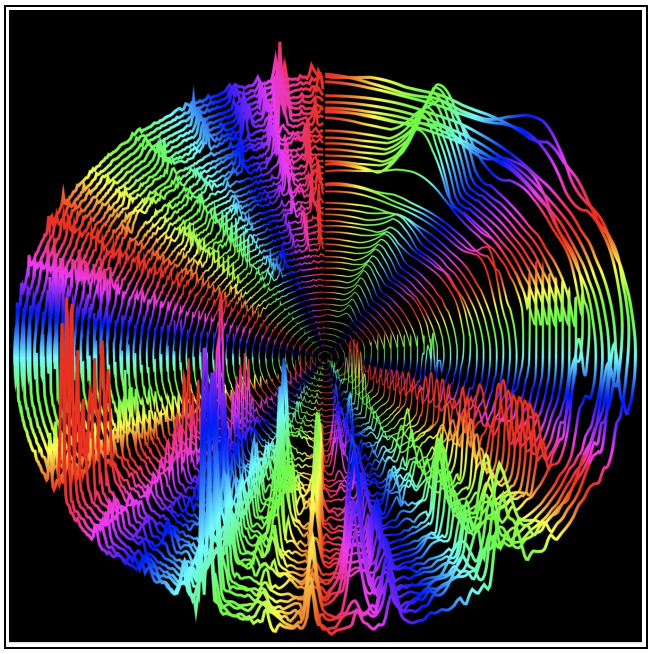
In addition to the current 864-point muSpectrum, we also render the previous 48 muSpectra. Hence, the above figure shows a total of 864 * 48 = 41,472 data points. We use two for() loops. The inner loop counts through the 6 octaves (0 <= oct < 6). The outer loop counts through the 48 spectra stored in the muSpecHistory[] buffer (0 <= hist < 48).

The different octaves are rendered in different vivid colors - hence the name RainbowOAS.

If the Option button is clicked, then (as shown in the right figure above) each octave of the muSpectrum uses a color gradient corresponding to the standard “hue” range – cycling through red, orange, yellow, green, cyan, blue, magenta, and back to red.

Rainbow Ellipse



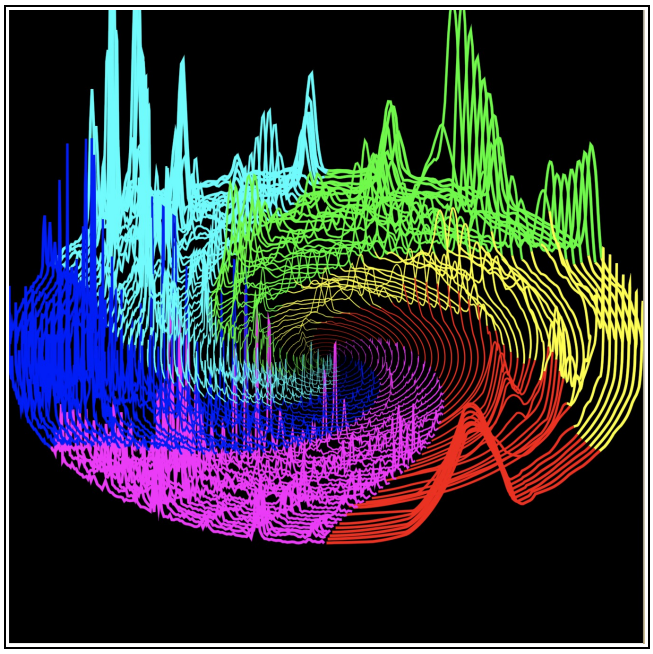


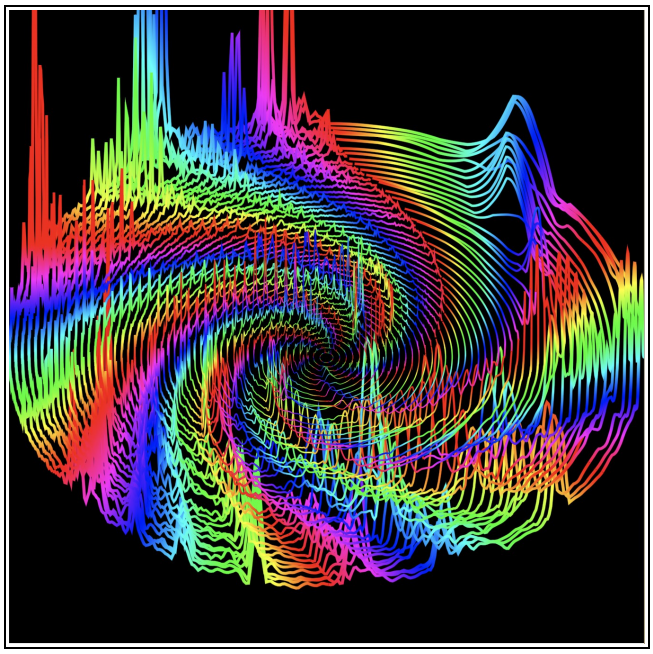
The RainbowEllipse visualization is a fun entertaining visualization. It is a "dancing light show" choreographed to the music using the live muSpectrum. It shows a 6-octave muSpectrum wrapped around an ellipse. It stores the spectral history (i.e., the most recent 48 muSpectra) in a buffer and uses iterative scaling to make the older spectral values drift into the center. That is, the most recent muSpectrum is rendered in the outermost ellipse, and each chronologically-older muSpectrum is rendered in the adjacent inner ellipse.

Each muSpectrum is rendered clockwise starting at the twelve o'clock position. Each ellipse is rendered in a different color, and these colors change dynamically with time so that a given muSpectrum remains the same color as it propagates from the outer to the inner ellipse.

If the Option button is clicked, then (as shown in the right figure above) each octave of the muSpectrum uses a color gradient corresponding to the standard “hue” range – cycling through red, orange, yellow, green, cyan, blue, magenta, and back to red.

Spinning Ellipse

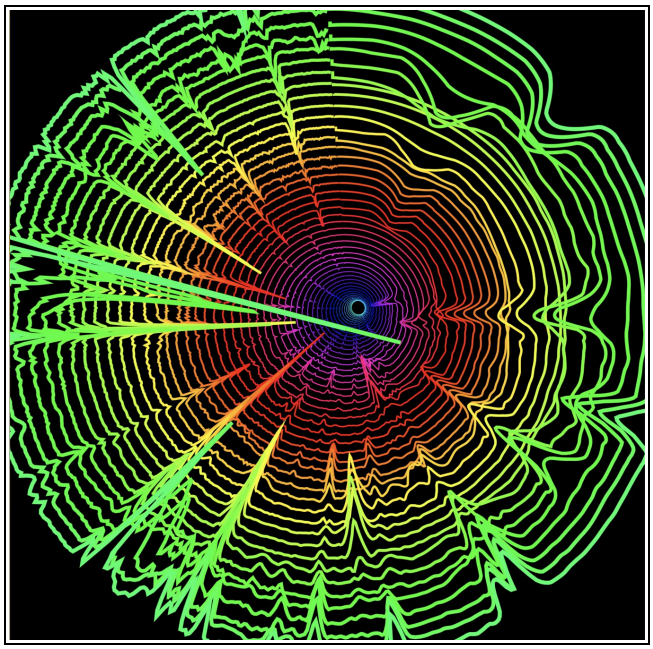


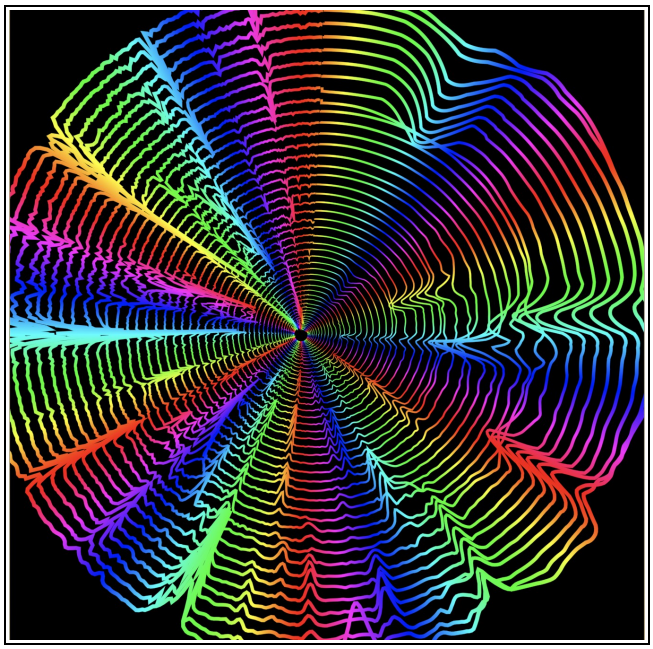


The SpinningEllipse visualization makes the RainbowEllipse more dynamic by making it spin. Also, an angular offset is applied to each muSpectrum to make the inward drifting of the spectral history more interesting.

Again, this visualization is intended as entertainment. I don't see any musical benefit from it.

Out of the Rabbit Hole





This “Out of the Rabbit Hole” visualization renders the time-history of the music's spectrum across a sequence of ellipses. The spectral data propagates from the outer ellipse to the inner ellipse – giving a psychological effect of the viewer coming out of the “rabbit hole”.

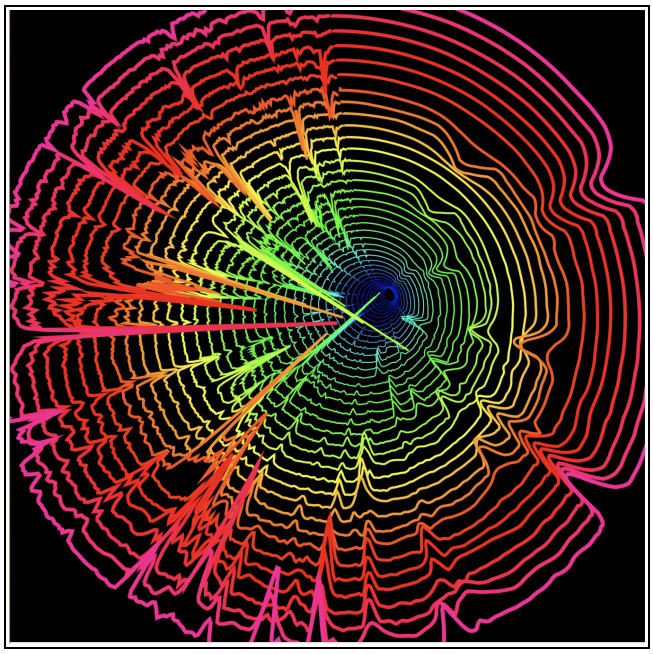
The following activities are performed during each rendering cycle:
1. The 6-octave muSpectrum is computed. It extends from C1 at 33 Hz to B6 at 1976 Hz. Each ellipse consists of 6 * 12 * 12 = 864 points.
2. These spectral values are written into a buffer memory.
3. The most recent 48 muSpectra are read from this buffer memory and rendered along the 48 concentric ellipses. Each muSpectrum is rendered clockwise starting at the twelve o'clock position.

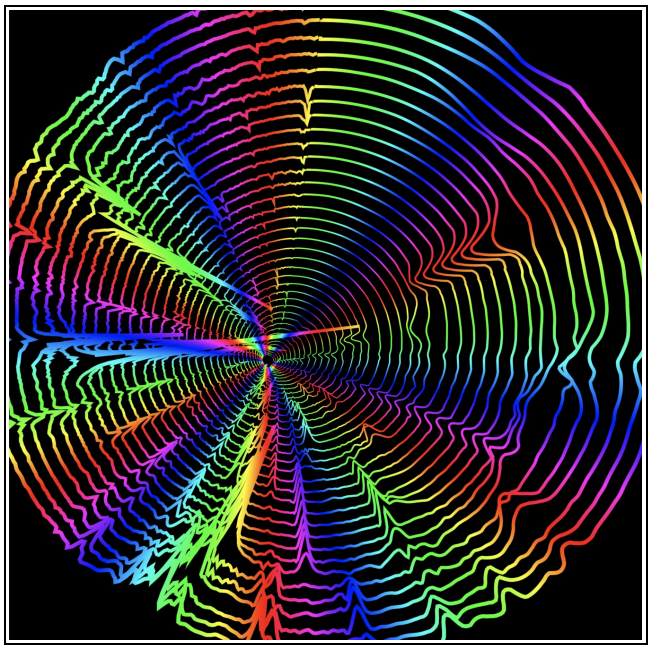
For a pleasing dynamic effect, the center-point for the ellipses varies with time.

Each ellipse is rendered in a different color, and these colors change dynamically with time so that a given muSpectrum remains the same color as it propagates from the outer to the inner ellipse.

If the Option button is clicked, then (as shown in the right figure above) we impose an angular gradient across the pane – such that each octave of the muSpectrum has a color gradient corresponding to the standard “hue” range – cycling through red, orange, yellow, green, cyan, blue, magenta, and back to red. The really observant user may notice that this effect is imperfect: The angular gradient is applied statically to the entire pane, whereas the ellipse centers move dynamically

Down the Rabbit Hole





This “Down the Rabbit Hole” visualization renders the time-history of the music's spectrum across a sequence of ellipses. The spectral data propagates from the inner ellipse to the outer ellipse – giving a psychological effect of the viewer falling into the “rabbit hole”.

The following activities are performed during each rendering cycle:
1. The 6-octave muSpectrum is computed. It extends from C1 at 33 Hz to B6 at 1976 Hz. Each ellipse consists of 6 * 12 * 12 = 864 points.
2. These spectral values are written into a buffer memory.
3. The most recent 32 muSpectra are read from this buffer memory and rendered along the outermost 32 of 40 concentric ellipses. Each muSpectrum is rendered clockwise starting at the twelve o'clock position.

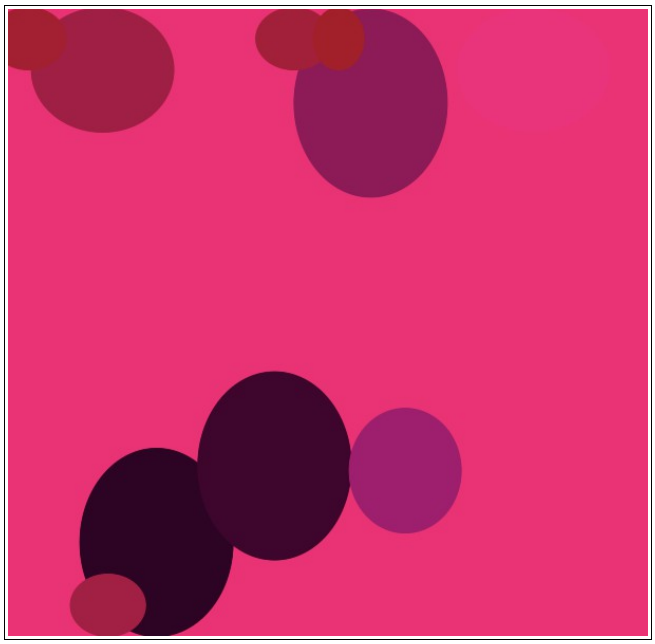
The reason for the peculiar arithmetic of this last activity is:
With a frame rate of 10 fps, and a buffer memory storing the previous 40 spectra, the data is 4 seconds old by the time it reaches the outermost (biggest and boldest) circle. The casual observer might not realize that the wiggles she is seeing are related to the sound that she is hearing (since the sound is 4 seconds older than the boldest wiggle). It looks better if we store only the previous 32 spectra and render them on the outermost 32 of the 40 circles. (The innermost 8 circles render a scaled-down version of the newest data rendered on circle number 9.) By repeating the newest muSpectrum 8 times, we increase it's emphasis and hopefully the casual viewer associates it with the audible sound she is hearing.

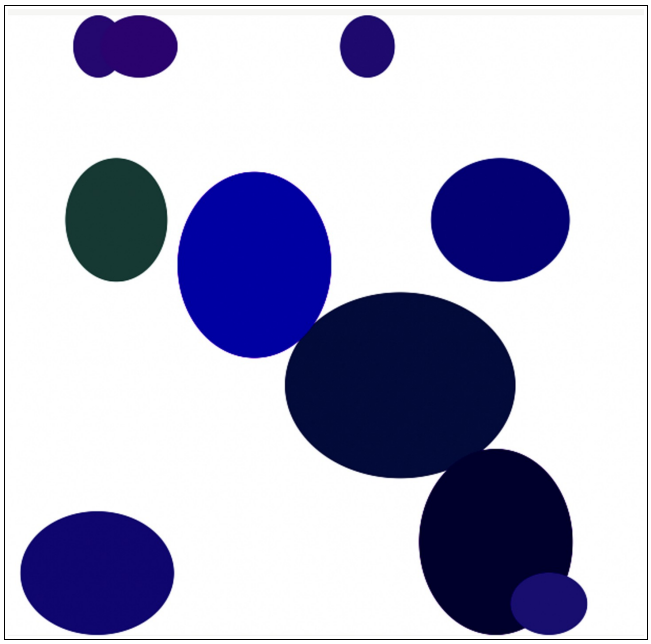
For a pleasing dynamic effect, the center-point for the ellipses varies with time.

Each ellipse is rendered in a different color, and these colors change dynamically with time so that a given muSpectrum remains the same color as it propagates from the innermost to the outermost ellipse.

If the Option button is clicked, then (as shown in the right figure above) we impose an angular gradient across the pane – such that each octave of the muSpectrum has a color gradient corresponding to the standard “hue” range – cycling through red, orange, yellow, green, cyan, blue, magenta, and back to red. The really observant user may notice that this effect is imperfect: The angular gradient is applied statically to the entire pane, whereas the ellipse centers move dynamically.

Lava Lamp





The Lava Lamp visualization is still a work-in-progress. Consider the version included here as my first attempt, with the desire to improve it as my experimentation continues. It is meant to imitate a traditional physical lava lamp that was popular back in the 1950s. Looking at the liquid blobs (bubbles of liquid) slowly morphing their shape and moving up and down was considered to be very relaxing.

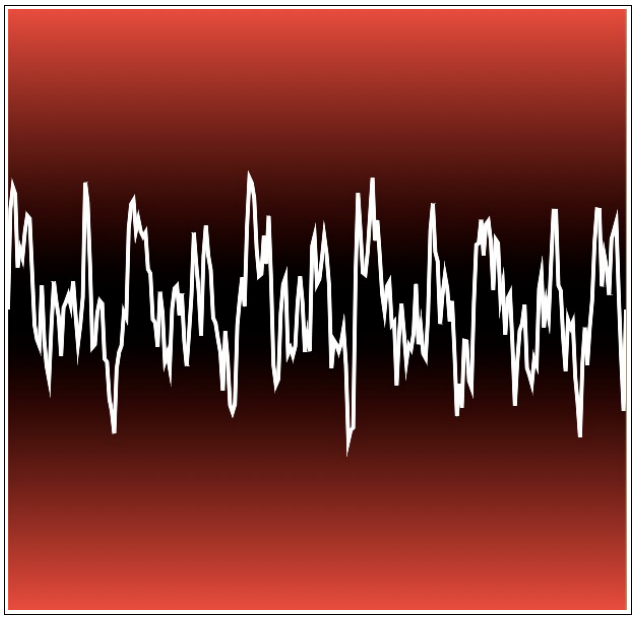
This particular implementation was inspired by Alex Dremov's "MorphingShapes" Swift package at
https://github.com/AlexRoar/MorphingShapes
as described in his article at
https://alexdremov.me/swiftui-advanced-animation/ .

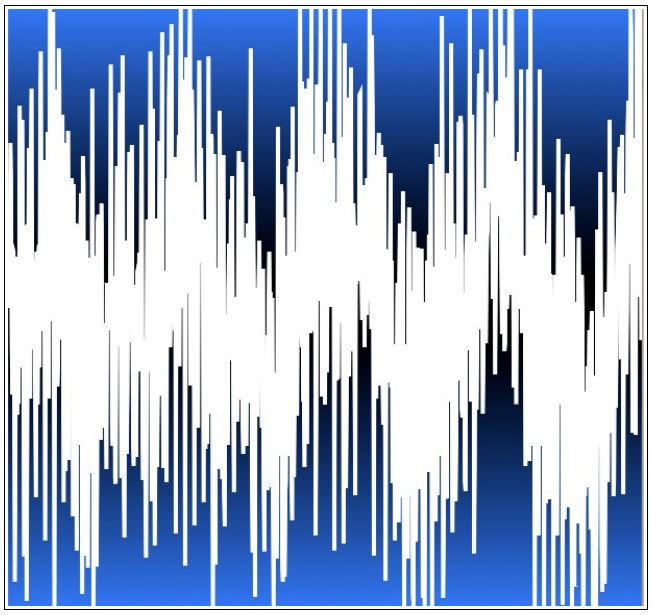
My initial desire was to have each blob morph it's shape corresponding to the loudness of each of the sixteen loudest spectral peaks within each frame of the music being played. But this proved overly computation- and graphics-intensive.

I settled on the idea of having ten blobs subdivided into three different sizes. Each blob is allowed to slowly oscillate from top to bottom randomly, and each blob is allowed to slowly move from side to side randomly. The blobs are actually ellipses – with the major- and minor-axes slowly oscillating in time. The color and brightness of each blob changes related to the strength of the spectrum in each of six octaves.

Also, the color of the background slowly change with time. If the OptionOn button is pressed, then the background color becomes either black or white (depending on the setting of the Dark/Light button).

Superposition





Some previous versions of MuVis have contained an “Oscilloscope” visualization that emulates the functionality of a physical piece of test equipment (called an "oscilloscope") used by electronic engineers to visualize a signal being tested. Typically, time is rendered on the horizontal axis and the voltage of the signal being tested on the vertical axis. If the signal contains a recurrent periodic pattern, then a "trigger" is used to start each horizontal time sweep whenever the signal voltage crosses a certain threshold. If the signal is perfectly periodic, this results in a stationary visual display of the waveform of the signal. Unfortunately, interesting music is not “perfectly periodic”, and results in rapidly time-varying highly-dynamic displays of the waveform. I found these Oscilloscope visualizations to be too chaotic to be informative – so I have not included them in recent releases of the MuVis app.

The Superposition visualization attempts to mimic an Oscilloscope but artificially makes all waveforms have the same phase, so the display is more stable and informative.

First, we measure the amplitudes and bin numbers of the 16 loudest spectral peaks of each frame of audio data. Then we generate 16 sinusoids with these amplitudes and frequencies - all with the same phase. (The origin of each sinusoid is the left-hand border of the rendering pane.) The visualization then sums these 16 sinusoids to provide their superposition.

The display renders 256 samples of each sinusoid – as shown on the left above. If the OptionOn button is pressed, then the display renders 1024 sample of each sinusoid – showing the superposition waveform over a longer duration – as shown on the right above.