Patrick Cincotta

Nik Bear Brown

CS 4300

12/18/2015

Realtime Music Visualization

My code-based research project’s theme is realtime music visualisation. It tries to encapsulate a vivid color style often used by projections and lasers at EDM events. The visualizer is composed of a few different parts that respond to the music being played at a given time. The code base is written in Java using Processing’s IDE and library.

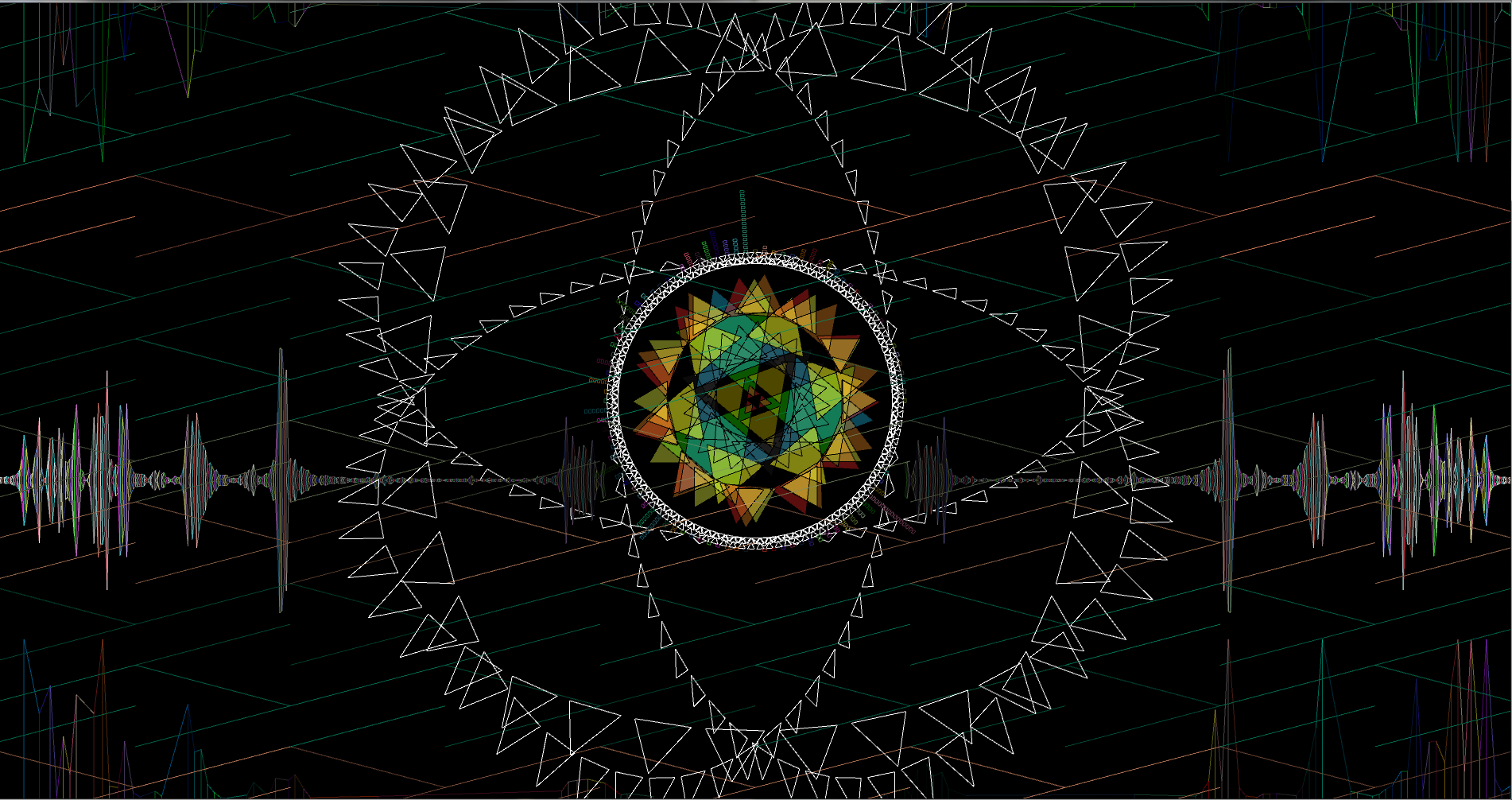
To get audio data I use Minim’s audio processing methods to get the audio stream into a buffer, run an FFT (Fast Fourier Transform) algorithm on both the left and right channels, combine the results into a mix channel array and keep the other two separate so that I can keep the option of using L/R/Mix at any point. New audio and display information is obtained in the logic thread runnable. That thread updates the audio buffer, updates global stats about the audio buffer, and sets the levels arrays used for equalizers in the scene.

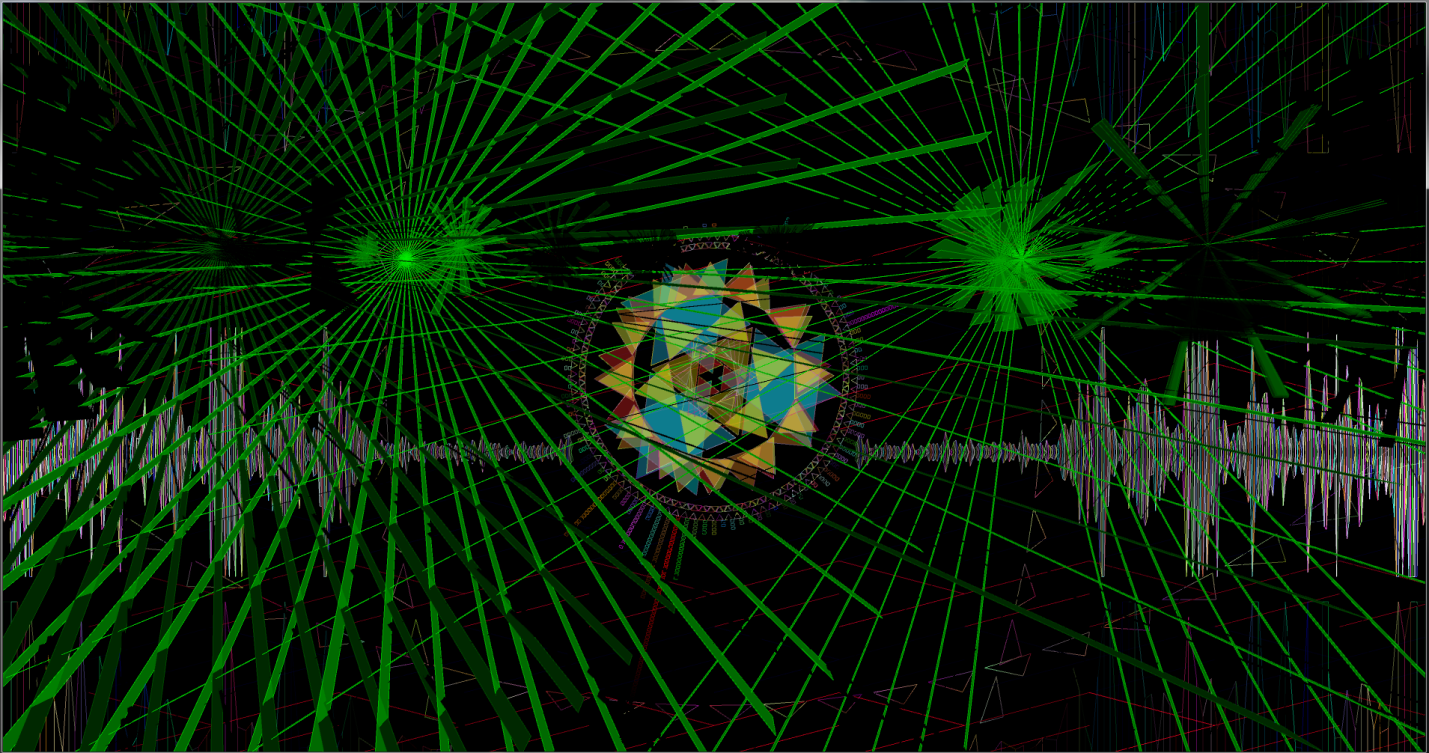
The center focus is made up of rotating rings of triangles of various colors/size, a double ring of small white triangles, equalizer bars going out radially, and a set of 4 3D sweeping rings. The equalizer bars can actually follow a different pattern other than the current audio buffer’s spectrum. By clicking on the scene the pattern number can be incremented through 5 different patterns. The max values of these patterns effects how background effects behave.

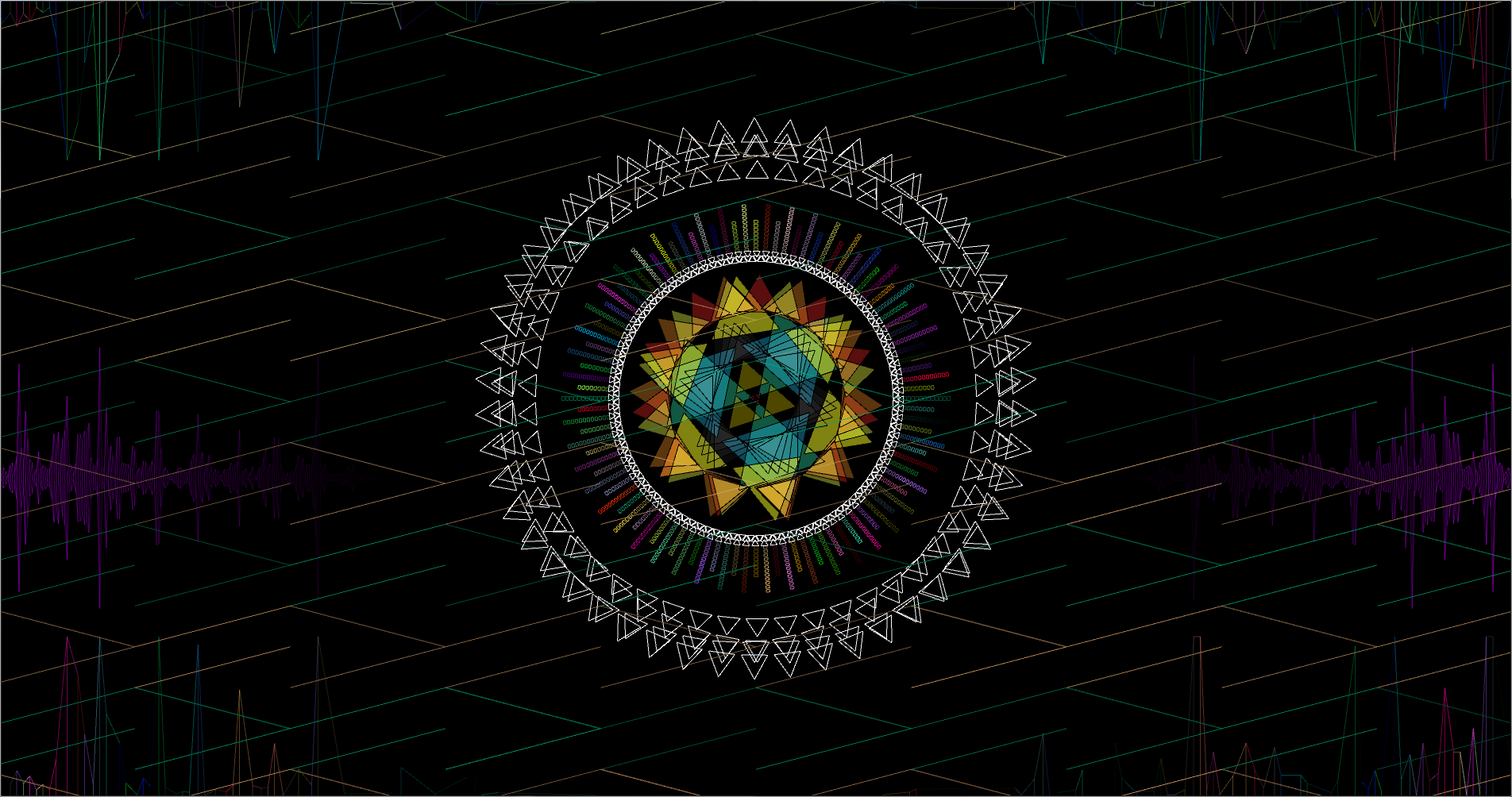
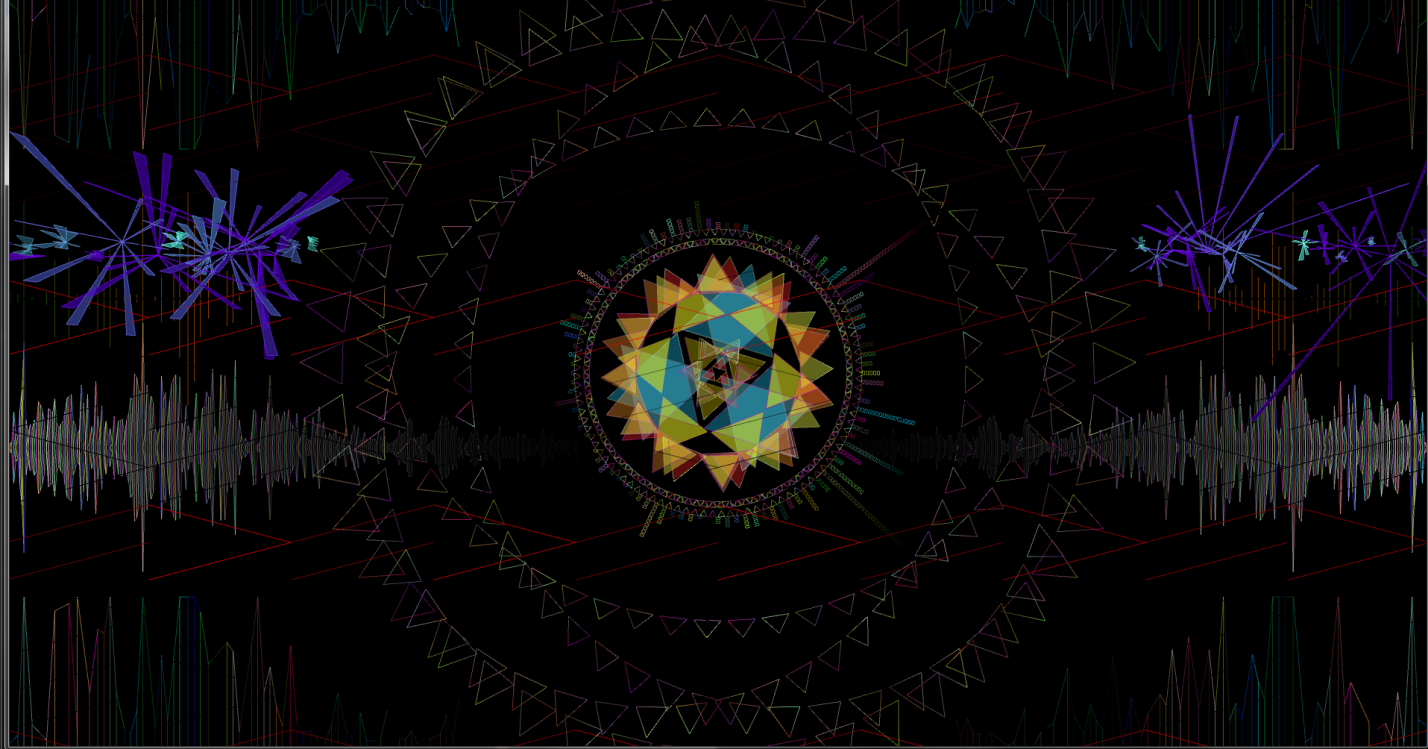
The background effects include a background design, a low-poly spectrogram, and a high density spectrogram display. The low poly points are decided upon by finding local peaks and valleys in the audio buffer. The higher density display uses every frequency it can until it starts to run into the center of the screen, then it fades out. The higher density display also changes its color set depending on the current max volume; if the volume is low it should either be completely faded out or using a solid color that rotates with time, and if the volume is high it should go into an ‘overdrive’ mode that uses a combination of bright colors that sweep back and forth across the lines over time. The background design uses a set pattern but has it’s color, scroll speed, and zoom depend on the track’s overal volume.

Finally, in the foreground there are two effects that are made to be ‘brighter’ than the rest of the scene. The first is the low density spectrogram that appears in orange/green above the high desnity spectrogram. It uses the same information as the low poly spectrogram but doesn’t draw connecting bridge lines and has some lower y values displacement to deceptively make it look reflective. Lastly there are the laser-like effects. it is produced by a function called aura\_spread. Like the other elements it’s paramaters are based on a function of time and overall volume in the track.

In order for all of this to be successful and look nice it needs to process new data quickly and pass it over to be drawn just as fast. I get this done by using Artur Fast’s threading framework (Fast). The logic thread does all data manipulation, the misc thread does framerate calculations and the draw thread is created by default by processing in the void draw () method. For low intensity tracks I suggest clocking on the scene to the 2nd or last pattern to enable the high/low density spectrograms a higher percent of the time. Otherwise, crank up the volume for some interesting effects and please enjoy responsibly!

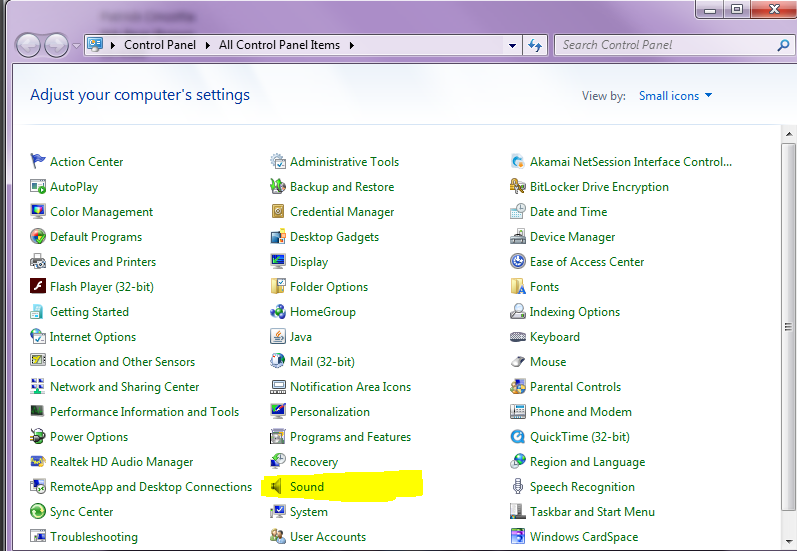




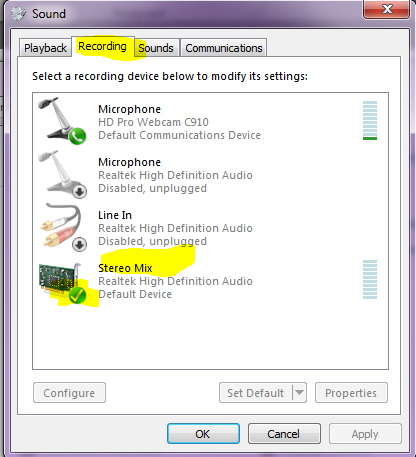


To get the visualizer to respond to music:

1. Open up sound from within control panel



1. Go to the recording tab and set Stereo Mix as the default device or use a microphone.



1. Now run the program with music on.

References

Fast, Artur. "Your Code Runs Slow." *Reddit*. N.p., 24 July 2015. Web. <https://www.reddit.com/r/processing/comments/3dypfk/your\_code\_runs\_slow\_and\_multi\_threading\_is\_too/>.