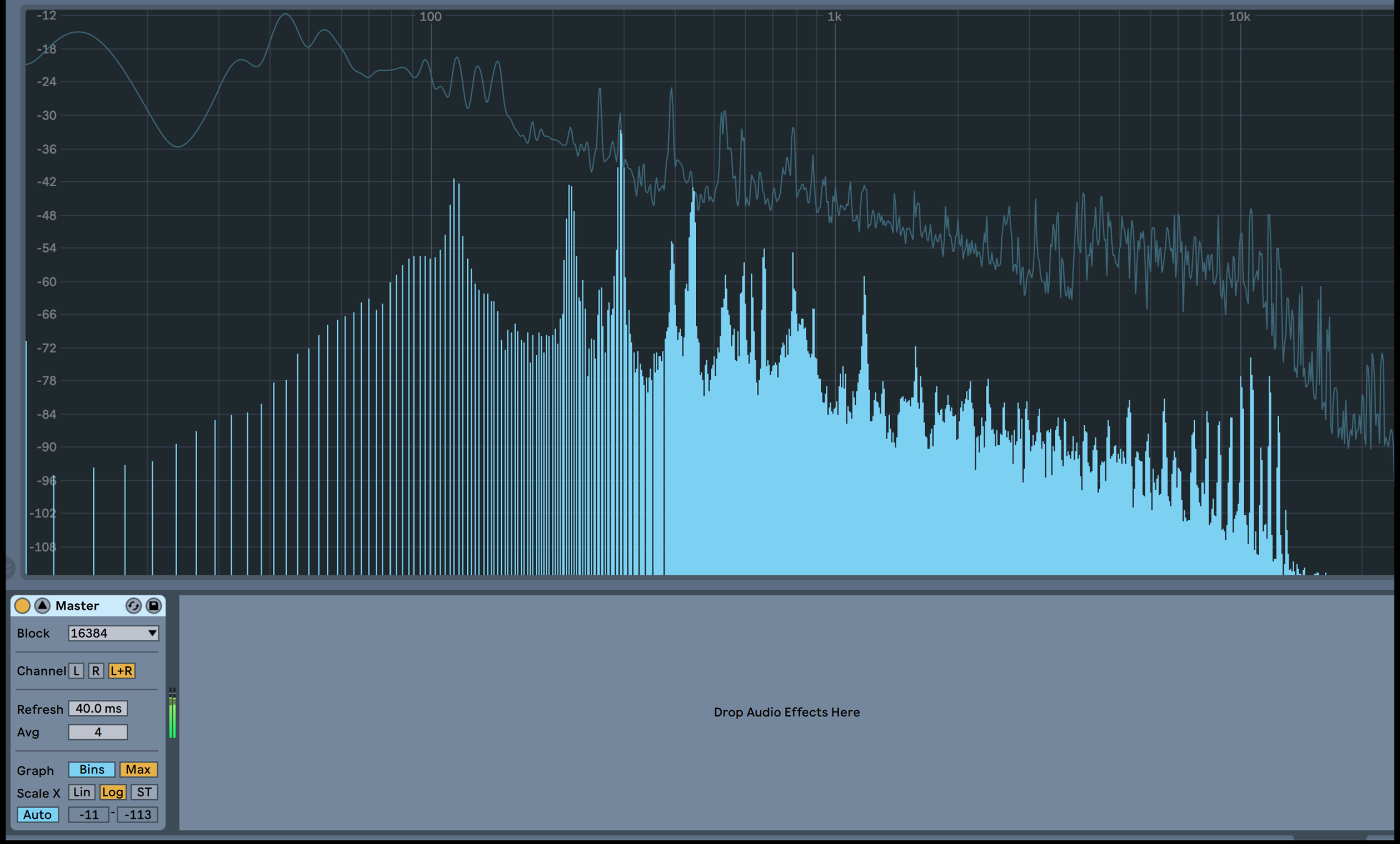
Open Source Music Visualization

[projectM]

Winamp

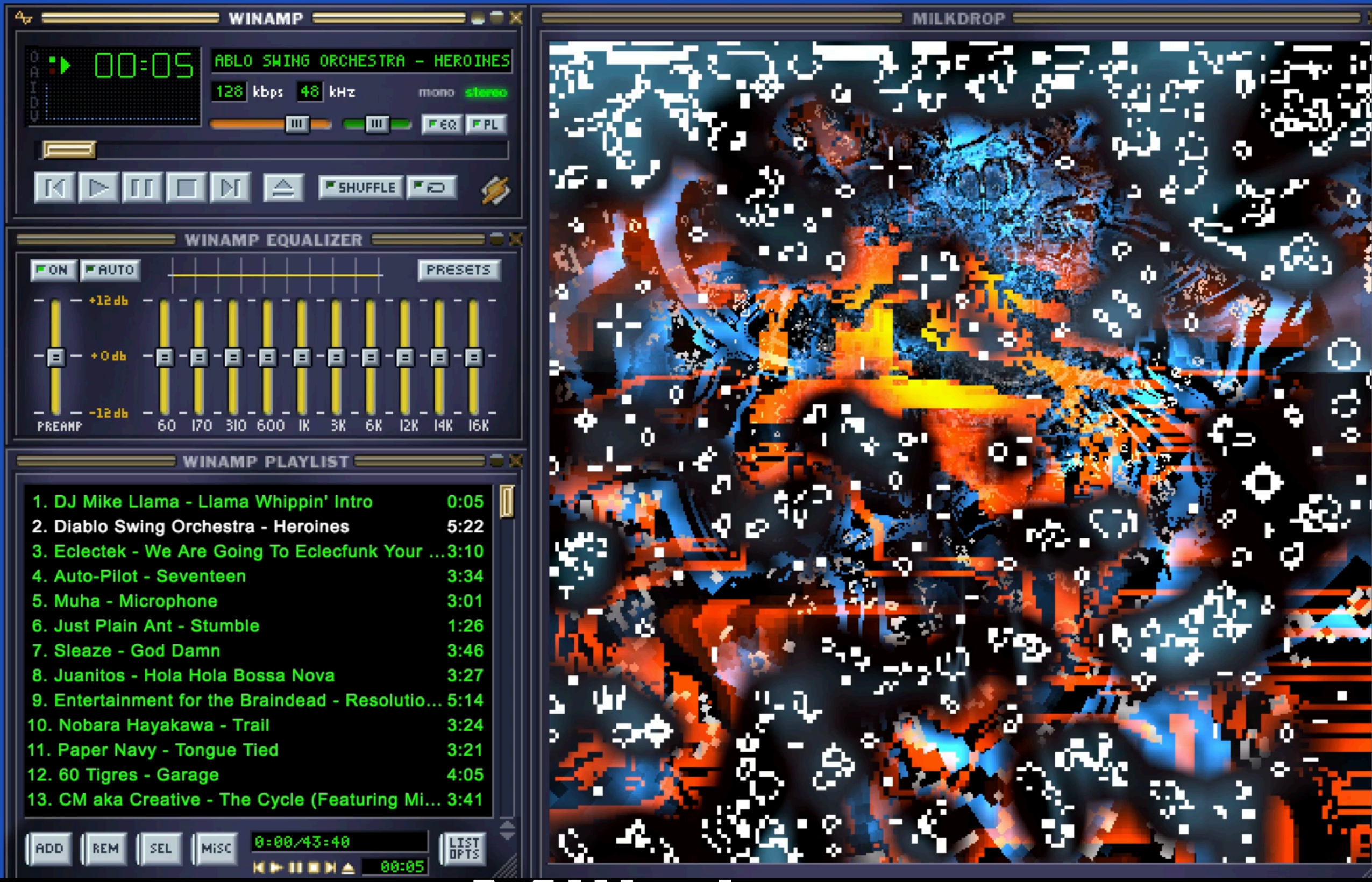
Visualilzer: Milkdrop

What is a music visualizer?



Fast Fourier transform

Measures amplitude of frequency bands


WINAMP
00:05
ABLO SWING ORCHESTRA - HEROINES
128 kbps 48 kHz
mono stereo
EQ PL
SHUFFLE
WINAMP EQUALIZER
ON AUTO
PRESETS
+12 db
+0 db
-12 db
PREAMP
60 170 310 600 1K 3K 6K 12K 14K 16K
WINAMP PLAYLIST
1. DJ Mike Llama - Llama Whippin' Intro 0:05
2. Diablo Swing Orchestra - Heroines 5:22
3. Eclectek - We Are Going To Eclecfunk Your ... 3:10
4. Auto-Pilot - Seventeen 3:34
5. Muha - Microphone 3:01
6. Just Plain Ant - Stumble 1:26
7. Sleaze - God Damn 3:46
8. Juanitos - Hola Hola Bossa Nova 3:27
9. Entertainment for the Braindead - Resolutio... 5:14
10. Nobara Hayakawa - Trail 3:24
11. Paper Navy - Tongue Tied 3:21
12. 60 Tigres - Garage 4:05
13. CM aka Creative - The Cycle (Featuring Mi... 3:41
ADD REM SEL MISC
0:00/43:40
00:05
LIST OPTS
MILKDROP

Milkdrop

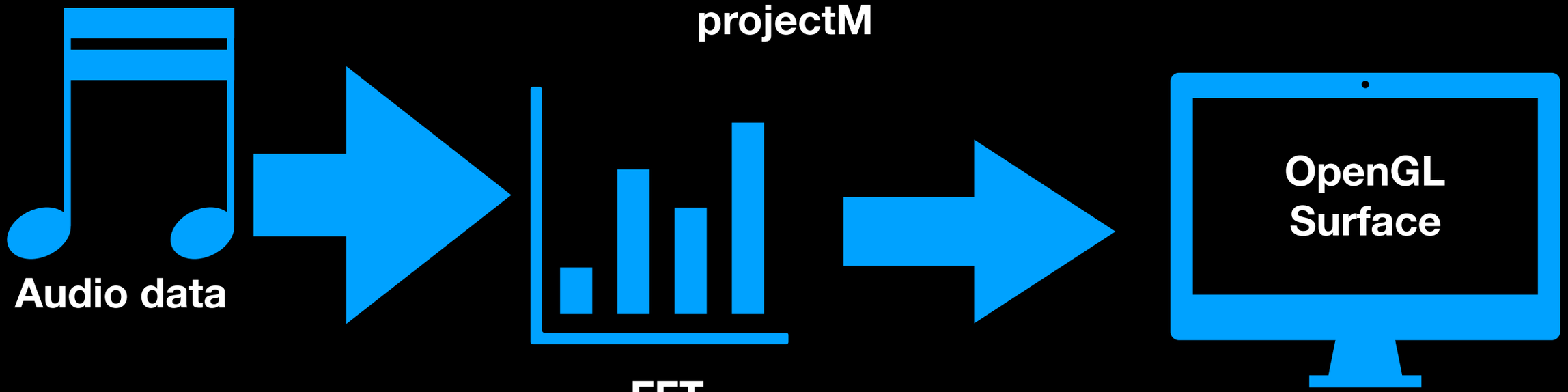
Milkdrop

- Windows only (win32 APIs)
- Assembly
- DirectX
- HLSL (DirectX shader language)

projectM

- Started in 2004
- Free/Libre Open Source Software (F/LOSS)
- Cross-platform (mac, windows, linux, BSD)
- Library
- Compatible with milkdrop presets

How it works



User-contributed content

martin - jellyfish dance.milk

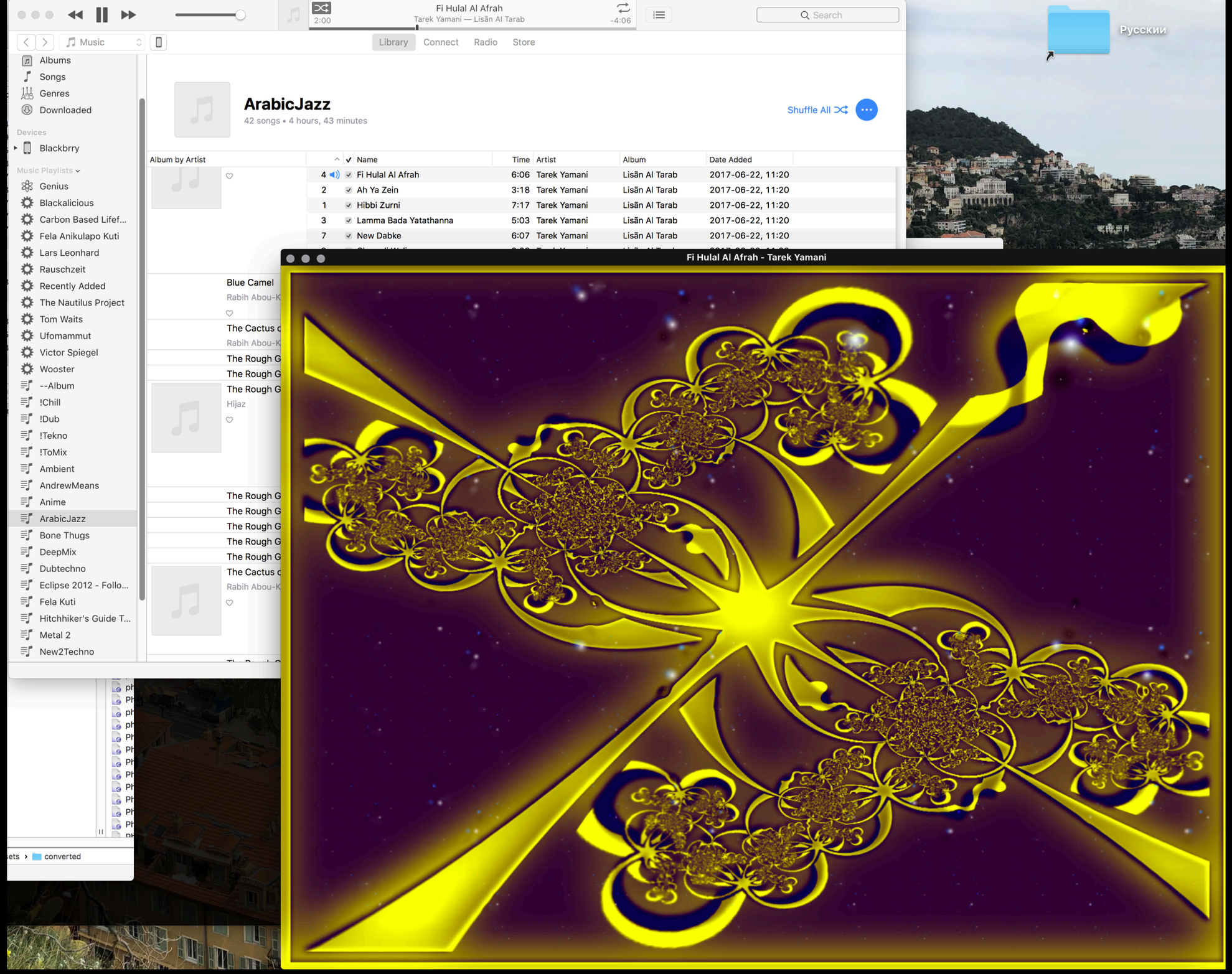
```
285 shapecode_3_g2=1.000000
286 shapecode_3_b2=1.000000
287 shapecode_3_a2=1.000000
288 shapecode_3_border_r=0.500000
289 shapecode_3_border_g=0.500000
290 shapecode_3_border_b=0.500000
291 shapecode_3_border_a=0.000000
292 per_frame_1=dec_med = pow (0.9, 30/f
293 per_frame_2=dec_slow = pow (0.99, 30/fps);
294 per_frame_3=beat = max (max (bass, mid), treb);
295 per_frame_4=avg = avg*dec_slow + beat*(1-dec_slow);
296 per_frame_5=is_beat = above(beat, .5+avg+peak) * above (time, t0+.2);
297 per_frame_6=t0 = is_beat*time + (1-is_beat)*t0;
298 per_frame_7=peak = is_beat * beat + (1-is_beat)*peak*dec_med;
299 per_frame_8=index = (index + is_beat) %8;
300 per_frame_9=index2 = (index2 + is_beat*bnot(index))%8;
301 per_frame_10=index3 = (index3 + is_beat*bnot(index)*bnot(index2))%3;
302 per_frame_11=
303 per_frame_12=q20 = avg;
304 per_frame_13=q21 = beat;
305 per_frame_14=q22 = peak;
306 per_frame_15=q23 = index;
307 per_frame_16=q24 = is_beat;
308 per_frame_17=q26 = bass + mid + treb + 1;
309 per_frame_18=q27 = index + 1;
310 per_frame_19=q28 = index2+1;
311 per_frame_20=
312 per_frame_21=k1 =  is_beat*equal(index%2,0);
313 per_frame_22=p1 =  k1*(p1+1) + (1-k1)*p1;
314 per_frame_23=p2 = dec_med * p2+ (1-dec_med)*p1;
315 per_frame_24=rott = p2 * 3.1416/4;
316 per_frame_25=
317 per_frame_26=q1 = cos(rott);
318 per_frame_27=q2 = sin(rott);
319 per_frame_28=q3 = -q2;
320 per_frame_29=q4 = q1;
```

Variables + functions:

sin() cos() time wave_r/g/b
bass mid treb dx dy rad min()
is_beat pow() fps bnot() max()
x y randomize() zoom rot…

```
per_frame_1=t = time*2.3;
per_frame_2=wave_x = wave_x + 0.350*( 0.70*sin(2.221*time) + 0.30*sin(1.821*time) );
per_frame_3=wave_y = wave_y + 0.350*( 0.30*sin(1.942*time) + 0.70*sin(2.522*time) );
per_frame_4=wave_r = wave_r + 0.790*( 0.60*sin(0.823*t) + 0.40*sin(0.916*t) );
per_frame_5=wave_g = wave_g + 0.790*( 0.60*sin(0.900*t) + 0.40*sin(1.023*t) );
per_frame_6=wave_b = wave_b + 0.790*( 0.60*sin(0.808*t) + 0.40*sin(0.949*t) );
per_frame_7=rot = rot + 0.010*( 0.60*sin(0.038*time) + 0.40*sin(0.054*time) );
per_frame_8=dx = dx + 0.002*( 0.60*sin(0.434*time) + 0.40*sin(0.277*time) );
per_frame_9=dy = dy + 0.002*( 0.60*sin(0.384*time) + 0.40*sin(0.477*time) );
warp_1=`shader_body
warp_2=`{
warp_3=`    // sample previous frame
warp_4=`    ret = tex2D( sampler_main, uv ).xyz;
warp_5=`
warp_6=`    // feather pen
warp_7=`    ret = max(ret, tex2D( sampler_main, uv + float2(1,0)*texsize.zw ).xyz);
warp_8=`    ret = max(ret, tex2D( sampler_main, uv - float2(1,0)*texsize.zw ).xyz);
warp_9=`
warp_10=`    // darken over time
warp_11=`    ret -= 0.035;
warp_12=`
warp_13=`    // add noise
warp_14=`    //float2 uv_noise = uv*texsize_noise_lq.zw*texsize.xy + rand_frame.xy;
warp_15=`    //ret += (tex2D(sampler_noise_lq, uv_noise)*2-1)*0.02;
warp_16=`}
comp_1=`
comp_2=`
comp_3=`
comp_4=`
comp_5=`shader_body
comp_6=`{
comp_7=`    uv = 0.05 + 0.9*uv;
comp_8=`    ret = tex2D(sampler_main, uv).xyz;
comp_9=`      // SUPER GLOW EDGES - looks awesome w/octopus
comp_10=`      float3 avg_col = GetBlur1(uv);
comp_11=`      ret = abs(avg_col - ret)*6;
comp_12=`    ret *= 1.333; // a little bit of overbright
comp_13=`}
```

```
115 class DLLEXPORT projectM
116 {
117 public:
118     static const int FLAG_NONE = 0;
119     static const int FLAG_DISABLE_PLAYLIST_LOAD = 1 << 0;
120
121     struct Settings {
122         int meshX;
123         int meshY;
124         int fps;
125         int textureSize;
126         int windowWidth;
127         int windowHeight;
128         std::string presetURL;
129         std::string titleFontURL;
130         std::string menuFontURL;
131         int smoothPresetDuration;
132         int presetDuration;
133         float beatSensitivity;
134         bool aspectCorrection;
135         float easterEgg;
136         bool shuffleEnabled;
137         bool softCutRatingsEnabled;
138
139         Settings() :
140             meshX(32),
141             meshY(24),
142             fps(35),
143             textureSize(512),
144             windowWidth(512),
145             windowHeight(512),
146             smoothPresetDuration(10),
147             presetDuration(15),
```



How I got involved:
Fixing iTunes plugin

My Contributions

- Converted build system from `cmake` to GNU `autotools`
- Created GitHub repository
- Fixed iTunes plugin on OSX
- Made simple SDL application
- Articles
- OpenGL modernization (little bit)

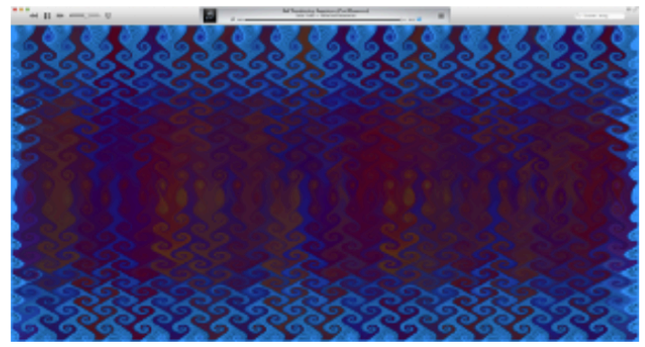
An introduction to projectM

March 28, 2018
This article was contributed by Mischa Spiegelmock

Many people have seen music visualizations before, whether in a music player on their computer, at a live concert, or possibly on a home stereo system. Those visualizations may have been generated using the open-source music-visualization software library that is part of projectM. Software-based abstract visualizers first appeared along with early MP3 music players as a sort of nifty thing to watch along with listening to your MP3s. One of the most powerful and innovative of these was a plugin for Winamp known as MilkDrop, which was developed by a Nullsoft (and later NVIDIA) employee named Ryan Geiss. The plugin was extensible by using visualization equation scripts (also known as "presets").

Sometime later, a project to implement a cross-platform, MilkDrop-compatible, open-source (LGPL v2.1) music visualizer began: projectM. The main focus of the project is a library (libprojectM) to perform visualizations on audio data in realtime—using the same user-contributed script files as MilkDrop—along with reference implementations for various applications and platforms. The project, which began in 2003 and was first released in 2004, is of interest to many for its creative and unique visuals, its use by media-player projects, and its interesting design and features. After years of development and contributions, the project stalled, but now there are efforts to rejuvenate and modernize the code.

How it works



LibprojectM is written in C++ and has a simple interface. The host application is responsible for creating an OpenGL context for the library to draw in, and then feeds in PCM audio data. From the audio data, the library extracts bass, mid, and treble amplitudes using the standard fast Fourier transform (FFT) that all audio visualizers use; it also attempts to perform beat detection to make the visuals synchronize with the music better. Each frame, the host application simply asks projectM to render its current visualization; it does that by drawing into the current OpenGL context. Here is the basic idea:

```
renderFrame() {
    glClear(...);

    projectM->pcm()->addPCMfloat(pcmData, 512);
    projectM->renderFrame();

    flipOpenGLBuffers(...);
}
```

When projectM renders a frame, it does so by passing the features extracted from the audio data to equations drawn from the currently selected MilkDrop preset file. There are two sets of equations (described in detail here), one set is evaluated per-frame and describes the shape, rotation, size, and colors of the waveform being drawn from the FFT data, and the other set is per-vertex equations for more complex transformations and deformations. These per-vertex equations are calculated for each point in a mesh (the points are the vertices), and then interpolated to the current screen display size, allowing less-powerful computers to reduce the number of vertices in order to sacrifice some accuracy for speed. MilkDrop preset files can also contain DirectX GPU shader programs, allowing for more complex programs and faster computation.

An introduction to projectM
Posted Mar 28, 2018 18:36 UTC (Wed) by **Inkane** (subscriber, #90843) [Link]

> Work was done recently to replace the problematic CMake-based build system with GNU Autotools.

That's a rather interesting statement. Well, whatever suits the current developers, but “coupled with its less-ubiquitous status” indicates a rather Unix centric view – on Windows, autotools is not exactly fun to use, cygwin non-withstanding.

Reply to this comment

An introduction to projectM
Posted Mar 28, 2018 18:49 UTC (Wed) by **boudewijn** (subscriber, #14185) [Link]

Yes... That doesn't inspire any confidence in the project. That's a serious case of blinkers.

Reply to this comment

An introduction to projectM
Posted Mar 29, 2018 9:09 UTC (Thu) by **revmischa** (subscriber, #74786) [Link]

Maybe I'm too stupid to figure out cmake

Reply to this comment

An introduction to projectM
Posted Mar 28, 2018 20:37 UTC (Wed) by **bahner** (subscriber, #35608) [Link]

What would work?

Reply to this comment

An introduction to projectM
Posted Mar 28, 2018 21:33 UTC (Wed) by **boudewijn** (subscriber, #14185) [Link]

CMake works fine... It's not perfect, but everything else is much, much worse. Not cross platform in any meaningful sense of the word (autotools), not mature (meson, qbs), derelict (wav, scons)...

Reply to this comment

An introduction to projectM
Posted Mar 28, 2018 22:18 UTC (Wed) by **rahulsundaram** (subscriber, #21946) [Link]

"It's not perfect, but everything else is much, much worse."

Meson is fine. A huge set of core components in the Linux ecosystem is relying on it now.

Reply to this comment

The Community

some fixes for itunes plugin support. still doesn't really work

 **revmischa** committed on Aug 4 ✗

itunes plugin works again

 **revmischa** committed on Aug 4 ✓

use bundled presets, skip config file. make self-contained

 **revmischa** committed on Aug 4

Commits on Aug 21, 2018

shaders fixes

deltaoscarmike committed on Aug 21

Add a fast preset switching test flag to check for shader compilation... ···

deltaoscarmike committed on Aug 21

Remove useless files

deltaoscarmike committed on Aug 21

Fix composite rendering glitches

deltaoscarmike committed on Aug 21

Fixing HLSLTranslator: ···

deltaoscarmike committed on Aug 21

Several HLSLTranslator fixes

deltaoscarmike committed on Aug 21

Complete HLSLTranslator matrix support

deltaoscarmike committed on Aug 21

Commits on Aug 22, 2018

Fix texture management

deltaoscarmike committed on Aug 22

Downstream

- Kodi / XBMC
- Helix music player
- Silverjuke (FOSS Jukebox)
- VLC (in progress)
- Debian / Ubuntu
- Arch
- Fedora
- Lots more I don’t know about

Proprietary Fork

- Permitted by LGPL (Lesser GNU Public License)
- iOS / Android

~End~

What open source topic or project do you want to talk about?