

An Audio Processing Library for Game Development in Flash

August 27th, 2009

Ray Migneco, Travis Doll, Jeff Scott, Youngmoo Kim,
Christian Hahn and Paul Diefenbach

Motivation

- Recent popularity of music-based games
 - ▶ *Guitar Hero, Rock Band and Dance, Dance Revolution*
 - ▶ Console platforms offer:
 - Rich graphics
 - Innovative control interfaces
 - Tight synchronization with audio processing
- Music-based games add an unprecedented amount of culture to the gaming experience
- Limitations of these games?
 - ▶ Most are bundled with pre-prepared tracks
 - What if you don't like the music?

Motivation

- Rise in popularity of web-based games
 - ▶ Wide availability of broadband connections
 - ▶ Improved client processing power
- Adobe Flash
 - ▶ Allows for rapid game development and deployment
 - ▶ Cross-platform support
 - ▶ Accommodates programmers of many skill levels

Music-centric Flash games

- *Super Crazy Guitar Maniac 2*
 - ▶ “Guitar Hero”-style game
 - Players press keys in response to derived beats
- *JamLegend*
 - ▶ Players can upload their own tracks
 - ▶ Limited audio processing
 - No control over instrument sound when notes “miss”
- *Music in Motion*
 - ▶ Side scrolling game: obstacles generated in response to music
 - ▶ Audio tracks are fixed and levels are preprocessed
- ...many more

Limiting factors

- Audio support
 - ▶ Previous versions restricted to just audio clip playback
 - ▶ Dynamic, buffer-based audio was recently added (ver 10)
- Processing limitations
 - ▶ ActionScript was not intended for computationally intensive algorithms
 - ▶ Some existing Flash-audio libraries authored in AS:
 - StandingWave - sound generation library
 - SoundTouch - time compression/expansion, pitch transposition
 - ActionScript Math Library - contains FFT functions

Adobe Alchemy

- What is Alchemy?
 - ▶ Allows C/C++ libraries to be integrated into Flash projects
 - C/C++ code is compiled to byte code, optimized for the ActionScript Virtual Machine (AVM2)
 - Minimal performance degradation on AVM2 from native compilation
 - ▶ Alchemy compiler generates a .swc file from C/C++ code, which is integrated into the Flash project
 - .swc : an archive file containing components and resources representing the C/C++ library

Alchemy performance

FFT Computation Times for Web-based platforms (msec)

	FFT Size						
Target	16384	8192	4096	2048	1024	512	256

- FFT computation time
 - ▶ Core function in DSP analysis/synthesis algorithms
 - ▶ 10,000 iterations for each size on each platform
 - ▶ Average elapsed times reported

Alchemy performance

FFT Computation Times for Web-based platforms (msec)

Target	FFT Size						
	16384	8192	4096	2048	1024	512	256
ActionScript (AS2 Math)	96.377	45.157	20.818	9.276	4.460	2.041	0.925

- FFT computation time
 - ▶ Core function in DSP analysis/synthesis algorithms
 - ▶ 10,000 iterations for each size on each platform
 - ▶ Average elapsed times reported

Alchemy performance

FFT Computation Times for Web-based platforms (msec)

Target	FFT Size						
	16384	8192	4096	2048	1024	512	256
ActionScript (AS2 Math)	96.377	45.157	20.818	9.276	4.460	2.041	0.925
Java (JTransforms)	29.517	20.703	9.393	4.345	1.956	0.901	0.385

- FFT computation time
 - ▶ Core function in DSP analysis/synthesis algorithms
 - ▶ 10,000 iterations for each size on each platform
 - ▶ Average elapsed times reported

Alchemy performance

FFT Computation Times for Web-based platforms (msec)

Target	FFT Size						
	16384	8192	4096	2048	1024	512	256
ActionScript (AS2 Math)	96.377	45.157	20.818	9.276	4.460	2.041	0.925
Java (JTransforms)	29.517	20.703	9.393	4.345	1.956	0.901	0.385
C (Alchemy)	3.009	1.371	0.628	0.297	0.139	0.067	0.034

- FFT computation time
 - ▶ Core function in DSP analysis/synthesis algorithms
 - ▶ 10,000 iterations for each size on each platform
 - ▶ Average elapsed times reported

The goal

- A fast, versatile and open source audio processing library for Flash game development
- Requirements:
 - ▶ Optimized for maximum performance
 - ▶ Flexibility for developers
 - ▶ Ease of implementation
- The result:
 - ▶ **ALF** - Audio processing Library for Flash

Architecture

- Alchemy is used to compile a .swc for a C/C++ DSP-based library
- DSP Audio Toolkit for Flash (DATF)
 - ▶ AS3-based wrapper for .swc file
- ALF is the top layer of abstraction
 - ▶ AS3 wrapper for DATF
- Open-source flexibility at all levels
- Varying developer interaction at each level

C/C++ DSP Library

Alchemy Compiler

Architecture

- Alchemy is used to compile a .swc for a C/C++ DSP-based library
- DSP Audio Toolkit for Flash (DATF)
 - ▶ AS3-based wrapper for .swc file
- ALF is the top layer of abstraction
 - ▶ AS3 wrapper for DATF
- Open-source flexibility at all levels
- Varying developer interaction at each level



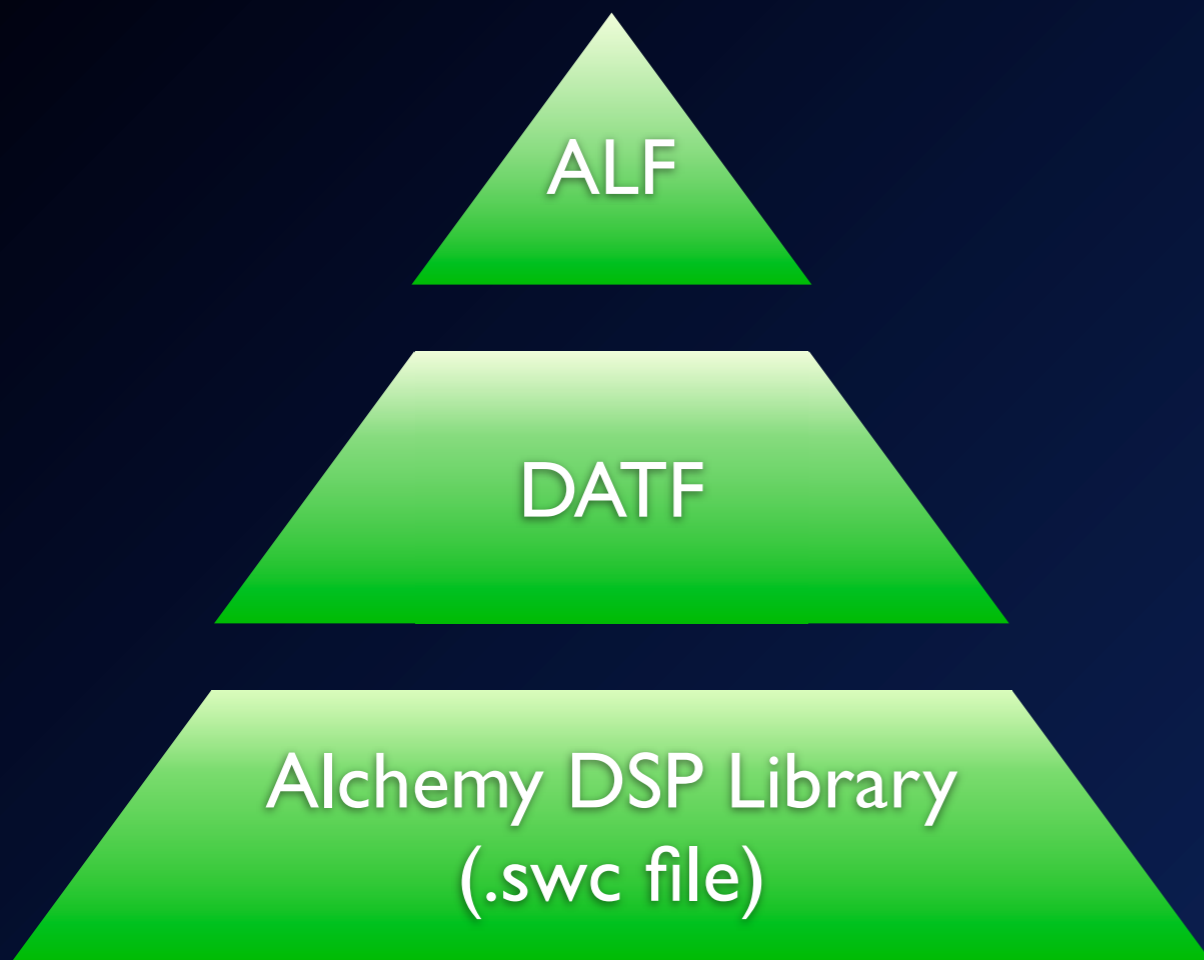
Alchemy DSP Library
(.swc file)

Architecture

- Alchemy is used to compile a .swc for a C/C++ DSP-based library
- DSP Audio Toolkit for Flash (DATF)
 - ▶ AS3-based wrapper for .swc file



Architecture



- Alchemy is used to compile a .swc for a C/C++ DSP-based library
- DSP Audio Toolkit for Flash (DATF)
 - ▶ AS3-based wrapper for .swc file
- ALF is the top layer of abstraction
 - ▶ AS3 wrapper for DATF
- Open-source flexibility at all levels
- Varying developer interaction at each level

Alchemy Layer

Alchemy DSP Library

Alchemy Layer

- Foundation of audio processing library
 - ▶ Alchemy compiled C/C++ code
- Contains:
 - ▶ Audio buffers
 - ▶ DSP library functions
- Developer responsibilities
 - ▶ Edit/add audio processing functions in C/C++
 - ▶ Compile .swc file and integrate



Alchemy DSP Library

DATF Layer

Alchemy DSP Library

DATF Layer

- DSP Audio Toolkit for Flash (DATF)
 - ▶ AS3-based code
- Interfaces with Alchemy DSP Library
 - ▶ Establishes shared AS3/Alchemy memory
- Developer responsibilities
 - ▶ Allocate audio buffers
 - ▶ Read and write buffers
 - ▶ Call DATF library functions



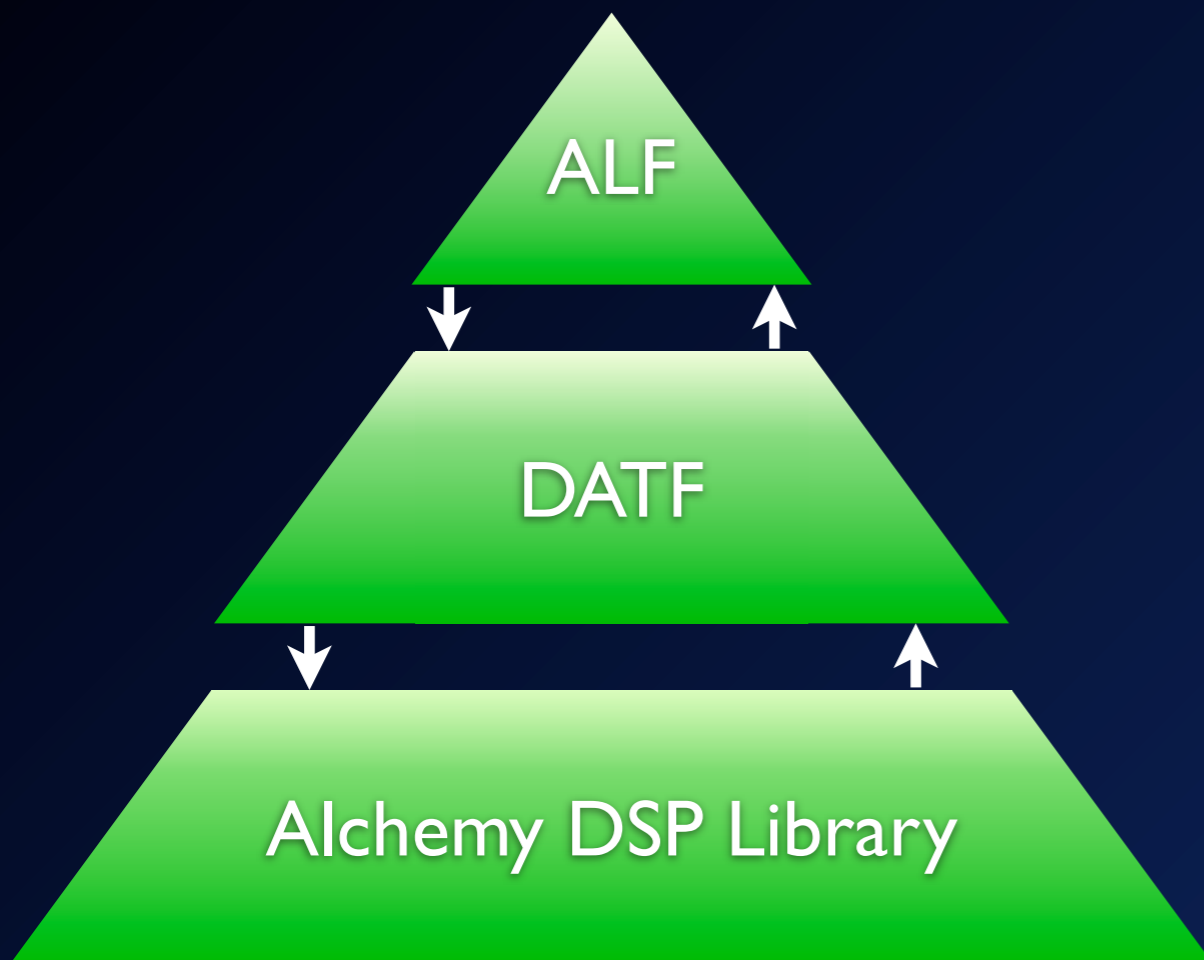
ALF Layer



DATF

Alchemy DSP Library

ALF Layer



- AS3-based code
- Developer specifies:
 - ▶ Audio file for analysis/playback
 - ▶ Desired ALF functions to use
 - ▶ Audio processing rate
- Simple interfacing
 - ▶ No DSP knowledge required
 - ▶ No memory management
 - ▶ No direct access to audio buffers

Audio Support

- Flash I/O limitations
 - ▶ Compatible with .mp3 and .wav file types
 - ▶ 22.05 kHz and 44.1 kHz sample rates
- Event listener is used to update the application when an audio frame has been computed

Available functions

- getSpectrum
- getIntensity
- getBandwidth
- getFlux
- getBrightness
- getHarmonics
- filter
- reverb

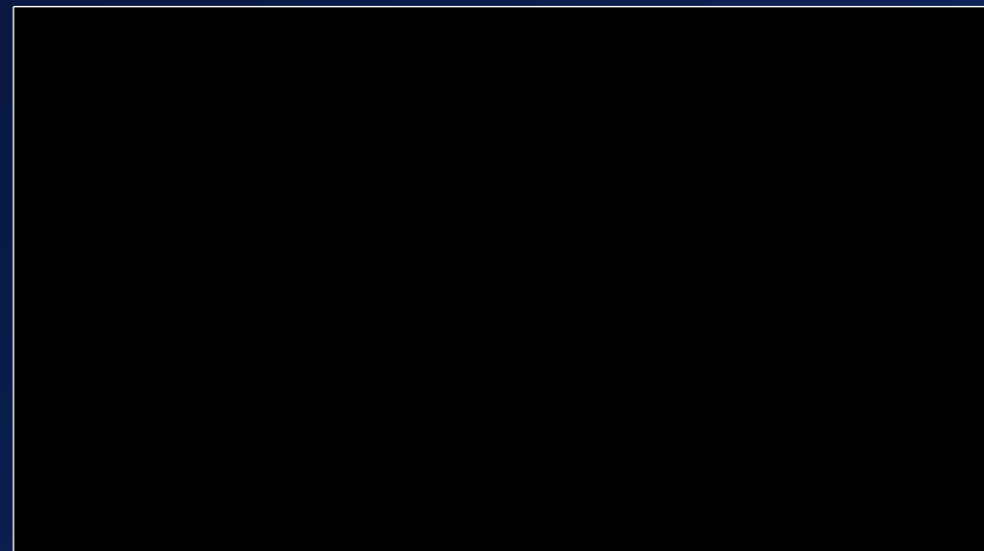
Available functions

- getSpectrum
- getIntensity
- getBandwidth
- getFlux
- getBrightness
- getHarmonics
- filter
- reverb





- Players upload .mp3 tracks from their music library
- Game environment is rendered in real-time based on game audio
 - ▶ Features are extracted using ALF and mapped to game parameters
- Development
 - ▶ Programmed and design with Adobe CS4 components
 - ▶ Deployed via Adobe Air 1.5: Flash for the Desktop
 - Avoids network latency for file uploads



Audio-driven environment



Audio-driven environment



Audio-driven environment



Audio-driven environment



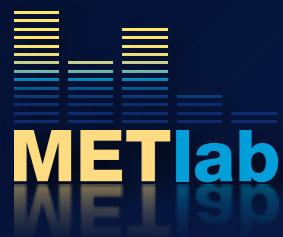
Audio-driven environment



Audio-driven environment



Pulse 2 Demo



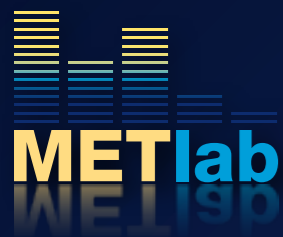
Conclusions

- Developed an efficient, open source, high-level interface for audio processing in Flash applications
 - ▶ Utilized computational benefits of Alchemy
 - ▶ Accommodates developers of varying skill levels
 - ▶ Capable of real-time processing
- Future work:
 - ▶ Expand the functional palette of ALF
 - Phase vocoder for time and/or pitch-scale modification
 - Additive synthesis methods for sound effect generation
 - Beat tracker

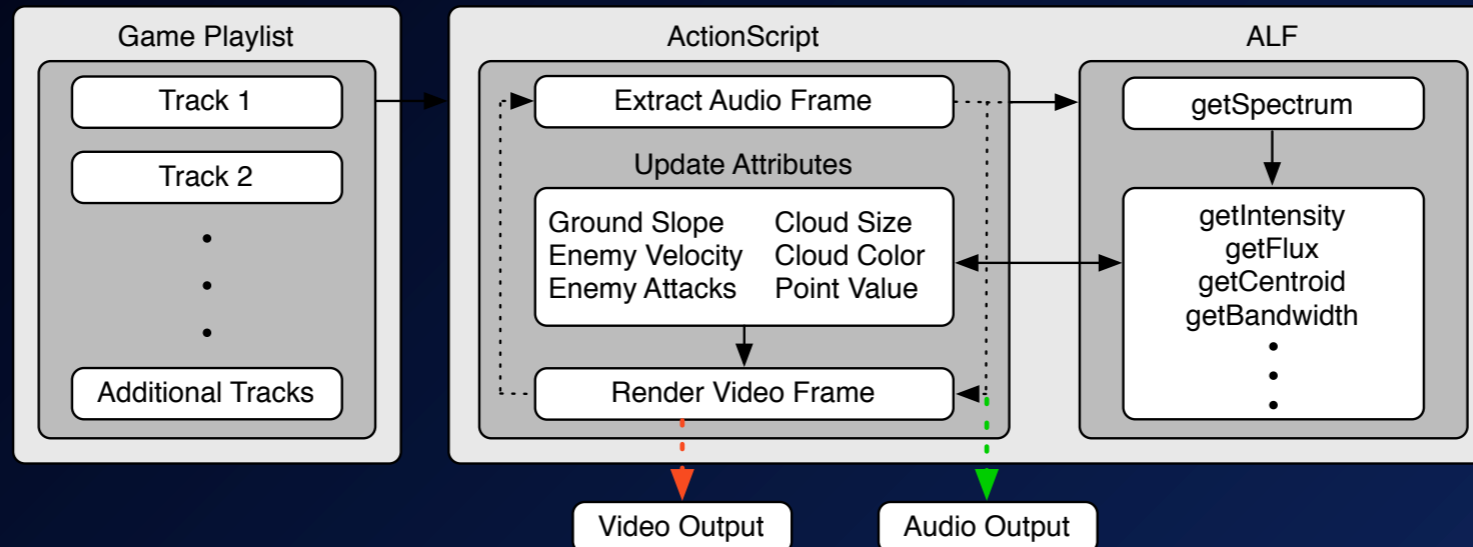
More info and downloads

- <http://music.ece.drexel.edu>
- Game links:
 - ▶ Pulse 2
 - ▶ Educational activities: *Tone Bender* and *Hide & Speak*
- ALF
 - ▶ Preview documentation
 - ▶ Sources coming soon

Questions?



Architecture



- show the game Architecture using ALF
- utilized functions, etc;
 - ▶ getFlux, getIntensity, getCentroid, getBandwidth

