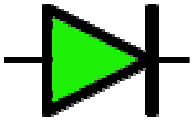
audioGraph

(under construction 12/4/11)



iOS Audio Processing Graph demonstration

"An **audio processing graph** is a Core Foundation–style opaque type, AUGraph, that you use to construct and manage an audio unit processing chain. A graph can leverage the capabilities of multiple audio units and multiple render callback functions, allowing you to create nearly any audio processing solution you can imagine" *–From Apple's Audio Unit Hosting Guide For iOS*

AudioGraph is a superset of Apple's MixerHost application.

Features include:

- Mono & stereo mic/line input
- Audio effects including:
  - Ring modulator
  - FFT passthrough using Accelerate vDSP framework.
  - Real-time pitch shifting and detection using
  - Simple variable speed delay using a ring buffer
  - Recursive moving average filter with variable number of points
  - Convolution example with variable filter cutoff frequency
- Stereo level meter
- Synthesizer example - sine wave with envelope generator
- iOS 5 features (from Chris Adamson) including:
  - MIDI sampler audio unit
  - file player audio unit
  - audio unit effects
- Runs on iPad, iPhone, and iPod-Touch
- Open source
- Music by Van Lawton
- Plus everything from MixerHost

Requirements

A device that runs iOS 5.x
Headphones.

Instructions

Launch the app and press Play.

Source code, documentation, support

Complete source code in Xcode project format is available from:

http://zerokidz.com/audiograph

Credits

Chris Adamson
Stefan Bernsee
Michael Tyson
Steven Smith
Everyone on the Apple Core-Audio mailing list
Various contributors to stackoverflow.com
Apple iOS developer program

Thank you.

keywords

core audio, iOS, audio units, audio processing graph, core midi, iPad, iPhone, iPod-Touch, MIDI, Sampler, FFT, Accelerate Framework, DSP, vdsp, objective-C, C, C++, audio, signal processing, digital filters, STFT, audio effects, convolution, open source, iOS 5, callback, streaming, pitch shifting, pitch detection.

System design and programming

Getting started

I had no intention to write this code. I was working on another project – trying to detect the audio frequency of a car engine. Soon my head was swimming with Core Audio, audio units, and processing graphs. There is no definitive set of instructions. Currently

the best resource is the work of Chris Adamson, including the draft of his upcoming "Core Audio" book.

The Apple developer resources are great – but they only lead you to the water's edge. The best resource is sample code that actually works.

If you're looking for answers, they're in the code. The source code for this project is available in Xcode 4 format. I recommend downloading it and running the audioGraph.xcodeproj (Xcode project) file in the main folder.

Core Audio programs are typically a hybrid of C, Objective-C and C++. My programming style is cut and paste. Some might call it slash and burn. In other words, don't expect anything you would find in a computer science book. For example, you will see at least three different methods of printing error messages to the console. Why? Code was copied from three different places. The goals here are:

- Make stuff work
- Try to understand why and write it down
- Optimize

You can learn a lot by removing pieces of code and seeing what breaks. My nephew took apart appliances and reassembled them using fewer parts. They would *usually* work just fine.

Please read the Apple Audio Unit Hosting Guide For IOS and any other material you can find by Googling "iOS Core Audio" – especially work by Chris Adamson. You may not understand it the first time. Unlike human relationships, eventually it will make sense.

Just in case nobody mentioned this: To actually run your applications on a device you need to become a member of the iOS developer program ($99). Sorry.

Recommend development setup

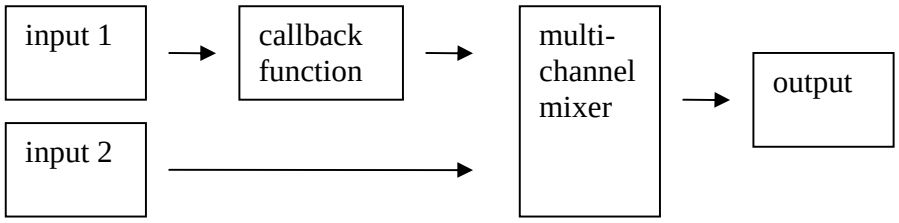
- Device running iOS 5.x
- Mac Computer running OS 10.6 or greater
- Xcode 4.x

The simulator is great but not really adequate for testing Core Audio. I recommend using the newest device you can find, with the most recent operating system. Another useful but non-essential item…

- iOS MIDI interface (or a WIFI MIDI app like Midi Touch)

Overview

Audiograph is a superset of the MixerHost sample code from Apple. MixerHost provides structure for processing audio at the sample level in callback functions.



This structure is called an audio processing graph. The components are Audio Units.

Callback functions pull audio from a source, processes it, and send it along. Processing typically includes things like:

- filtering
- effects
- analysis
- synthesis

Callback functions can also act as signal generators – synthesizing audio by producing sample data.

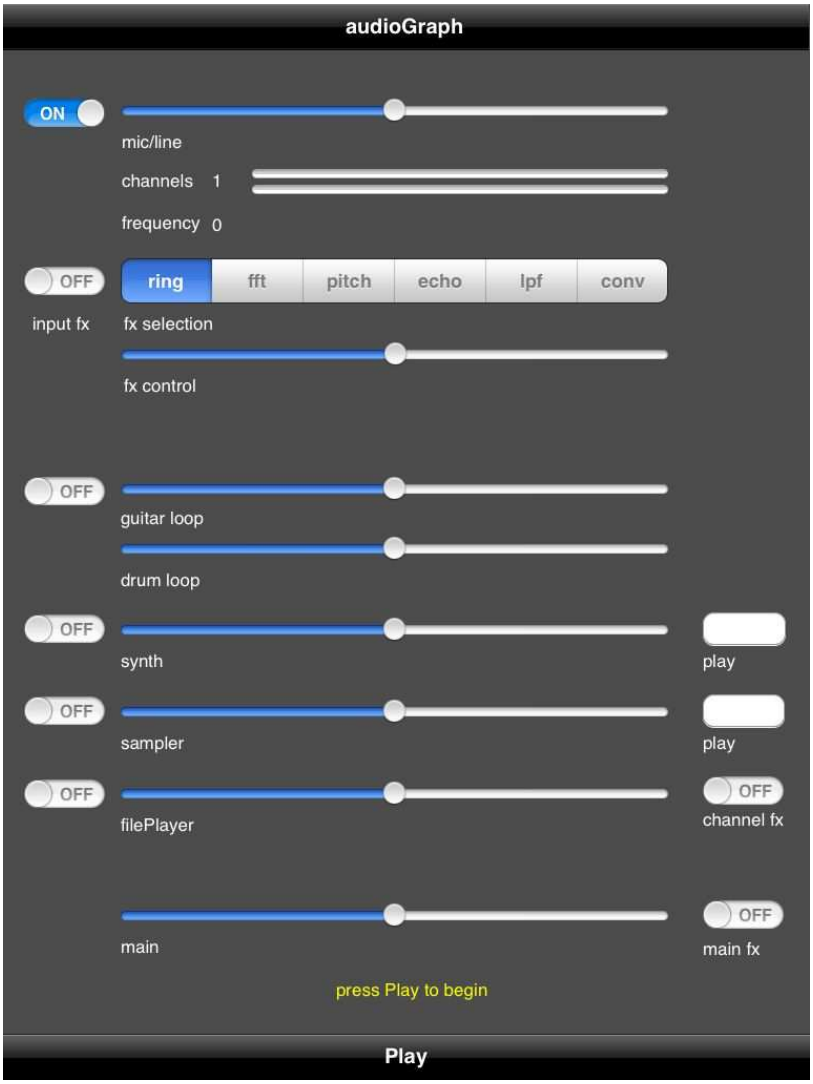
Prior to iOS5, audio units were limited to IO and mixing, and format conversion. New Audio Unit types added in iOS5 include:

- Effects (reverb, filtering, distortion, etc.,)
- A MIDI sampler
- A file player

Rather than explain the entire world of iOS Core Audio or get bogged down in signal processing we'll focus on specific tasks like reading stereo input data and how to do convolution. I'll point out what can go wrong and provide resources for further study.

Source files

The big picture



Insert schematic diagram

Infrastructure

Audio Session

Audio Units

Audio processing graph

Streams

Handling events and interruptions

Audio Unit Types

Remote IO Unit

Multichannel mixer unit

Effects Units

MIDI Sampler Unit

Fileplayer Unit

Audio callbacks

Sample type conversion

Multiple channels

Generating sound

Signal Processing – time domain

Ring Modulator

Simple delay

Ring buffers

Filters

Convolution

Signal Processing – frequency domain

FFT pass through

STFT, pitch shifting and detection