Adding AUv3 plugin support to a standalone iOS application

Nick Porcaro
Dr. Julius O. Smith III
Pat Scandalis

March 1, 2019 - CCRMA Open House - Stanford University

Background

Apple Audio Unit Version 3 (AUv3) App Extensions are a robust audio plugin format that can shield users from rogue or unstable plugins.

But the cost of this robustness can be large to developers. This talk will cover the challenges encountered when adding AUv3 support to GeoShred.

How are Audio Unit Extensions used?

- Used in AU Hosts as instrument track or an effect
- AU extensions can send MIDI to each other provided the host handles the routing (eg AUM, Audio Bus 3)
- Short demo: http://www.moforte.com/audemo
- More info on GeoShred: http://www.moforte.com/nicksfavs

Decide how to add AUv3 support

- Rewrite in portable way, but time consuming for a large system
- JUCE: Examples were too limited in early 2018
- No obvious benefit on Android because of performance limitations: ROLI MAQ: http://www.moforte.com/maq
- MacOS: Perhaps wait until cross platform UIKit
- Decided to port existing Objective-C code

README

- How app extensions work:
  http://www.moforte.com/appExtWorks

- App Extension programming guide:
  http://www.moforte.com/appExtGuide

- Apple WWDC video:
  http://www.moforte.com/wwdc2015

- AUv3 example project:
  http://www.moforte.com/appleAUv3Example

- Anything Michael Tyson writes:
  http://www.moforte.com/michaelTyson

- Comments in AudioToolBox/AUAudioUnit.h, CoreAudioKit/
  AUViewController.h

Make simplest possible example

- Swift example code manually converted to Objective-C
- Define a global state class
- Determine how to know whether running standalone or as an extension
- Entry point for audio unit is in main view controller, a subclass of AUViewController, which allocates an AUAudioUnit subclass
- Figure out how to wire up your existing signal processing and MIDI event processing into your AUAudioUnit subclass
- Use Autolayout if possible, for sure in the future

Retool the Xcode project

- Add audio unit app extension target to Xcode project, using the AUv3 example project as a guide
- Beware: Using a shared framework may degrade graphics performance. More on this later
- No read/write global data allowed, no singletons
- Some APIs not available, eg UIApplicationDelegate, see App Extension programming guide for more

Massive code refactoring

- Allocate all former read/write singletons in global state class
- Any object needing global state gets reference on the stack
- Remove singleton method definitions
- Build and fix the resulting errors, for a long time.
- Sometimes possible to do global editing, but not in general
- Separate UI from MIDI and signal processing, must run “headless” for out of real time rendering (eg: Garage Band Share).

Watch out!

- Eliminate read/write static variables otherwise there will be cross talk between AU instances
- Read-only static variables are OK
- One class variable needed to determine standalone or app extension (eg: AppGlobal.runningStandalone). Set to YES only in UIApplication delegate when launching
- No dynamic voice allocation or timers that are not based in the audio thread
- Be very careful not to introduce bugs, fix problems as you see them, or temporarily eliminate functionality with ifdefs
- Be patient and diligent, may take many days before linking

Bootstrap Sequence

- Initialization of render resources and view layout will likely be different standalone vs app extension

- Not all hosts handle view layout the same way - only knowable by experiment. Best to contact developer of errant hosts

- Come up with strategies to deal with these differences

- Autolayout would be best, but you might have to settle for affine transforms and related trickery

- Sample rate ultimately determined by the audio unit output bus.

internalRenderBlock

- Replaces the CoreAudio render callback from standalone world
- Audio and MIDI are received from the host via the internalRenderBlock, defined in the AUAudioUnit subclass
- MIDI read/write mechanisms are different, but standalone MIDI semantics (eg: GeoShred MIDI configurations) can still be used in app extension
- internalRenderBlock can conditionally process separate input and output buffers, so it can be configured as a filter or an instrument or both at the same time

internalRenderBlock flow

- Audio samples come in host
- Schedule MIDI out events which were queued from the main thread.
- Scheduled MIDI events appear to be throttled somehow. We saw this when writing the GeoShred panic sequence
- Process audio frame, by calling the signal processing compute function, which can be an Objective-C IMP style function pointer, or rewrite audio engine in C++
- Process MIDI in events which were queued using dispatch_async from the main thread
- Process scheduled MIDI out events

Host/audio unit responsibilities

- The audio unit can interpret MIDI input and output, but the host has to provide the actual I/O via CoreMIDI
- The audio unit has to respond to MIDI input and play it exactly the way it was recorded into the host
- Beat timing comes from host, the audio unit has to interpret rewind, and other shuttle controls by counting beats and guessing. Would prefer more formal protocol
- Host handles MIDI event recording, CoreMIDI setup, CoreAudio setup (session, input/output devices)
- Backing tracks can be handled by the host or by the audio unit.
- Audio unit could spooling samples out of backing track files in the signal processing compute function.

AU Parameter Tree

- Only way to persist data meta to your own file format
- Program number, bank number, and anything else you like
- Can define presets meta to the app extension’s native presets
- Would be nice if it was mutable, a bug report has been filed, until then parameter structure is flat, which is hard to manage if the app extension supports hundreds of parameters

File handling

- iTunes file sharing doesn't work the same way, use App Groups so the app extension and standalone app can share data
- Use cloud based file management, eg: UIDocumentPickerViewController
- Makes it very easy to move files around
- Move URL handling out UIApplication delegate

Graphics

- Implemented the GeoShred keyboard with OpenGLES, Core Animation and CoreGraphcs
- OpenGLES had the only acceptable performance
- Metal could work but the cost of retrofitting was too high
- In the future use Metal, as the new cross platform UIKit will not support OpenGL (at least not now)

Debug tracing

- Have a good system for debug tracing, with lots of options, especially because some hosts can be aggressive about killing unresponsive plugins.
- Sometimes cannot run in debugger because of real time event handling.
- Be able to copy trace file anywhere
- Come up with a way to indicate which instance is running in the trace output

Sharing a framework

- The Apple example code shows how you can share common code using a framework
- But performance can be impacted, perhaps because of inversion of control. I have not filed a radar on this yet.
- We had performance problems with graphics that were solved when flattening the project to not use a shared framework
- Determined by process of elimination
- A bug fix or work-around from Apple would be great

Testing

- Common hosts we used for testing: GarageBand, AUM, Audio Bus 3, Cubasis

- Not all host behave the same way in all situations, contact the host developer or come up with work arounds

- Note the various uses case for MIDI and audio routing that different hosts support

- Xcode is often hard to work with when single stepping from a host. GarageBand seems to be too aggressive about killing hosts that are slow to respond, either at load time or when single stepping. This often caused us to have resort printf style logging to debug.

- Some hosts, like AUM and AudioBus 3 are are not as picky about plugins taking a long time to respond.

Sneaky Tricks

- With slight modifications you have an AUAudioUnit subclass act both as a filter and an instrument

- All you need to do is take care in how you use the input and output busses and how the internalRenderCalback views the audio buffers

- Your extension can be both an instrument and MIDI processor if you do this in the plist for the extension

Sneaky Tricks (cont)

▼ NSExtension	Dictionary	(3 items)
▼ NSExtensionAttributes	Dictionary	(1 item)
▼ AudioComponents	Array	(2 items)
▼ Item 0	Dictionary	(9 items)
version	Number	67072
manufacturer	String	MFTE
sandboxSafe	Boolean	YES
factoryFunction	String	MainViewController
subtype	String	GSAU
description	String	GeoShred
type	String	aumu
name	String	MFTE: GeoShred
▶ tags	Array	(1 item)
▼ Item 1	Dictionary	(9 items)
version	Number	67072
manufacturer	String	MFTE
sandboxSafe	Boolean	YES
factoryFunction	String	MainViewController
subtype	String	GSAU
description	String	GeoShred
type	String	aumi
name	String	MFTE: GeoShred

An extension that is both an instrument (aumu) and MIDI processor (aumi)

Acknowledgements

- Thanks to the Apple CoreAudio team for their help and encouragement

- Also thanks to Geert Bevin of Moog Music and Michael Tyson of A Tasty Pixel

- Download this presentation here:

  http://www.moforte.com/2019CCRMAOpenHouse/AddingAUv3ToStandaloneApp.pdf

- Contact nick@ccrma.stanford.edu if you have any questions