

Macoun' I I

Core Audio und MIDI auf iOS

Boris Bügling / @NeoNacho

Motivation

- Unterwegs musizieren ist toll
- Apple sieht das genau so:
 - iOS 2.0 - Core Audio
 - iOS 4.2 - Core MIDI
 - iOS 5.0 - Neue APIs in Core Audio

Überblick

- Grundlagen, Einführung in Audio Units und Tipps
- Abspielen von Samples
- “Kochbuch” und Einlesen von MIDI Input
- Neuerungen in iOS 5
- Endergebnis: ein einfacher Synthesizer

Grundlagen

Digitale Audio Verarbeitung

- Digitales Signal: Messung zu diskreten Zeitpunkten
- PCM (Pulse Code Modulation)
- Sample == Amplitude des Signals zu einem bestimmten Zeitpunkt
- Sample-Rate doppelt so groß wie Frequenzbereich

Digitale Audio Verarbeitung

- Sample: Messwert einer Wveform
- Frame: Menge an Samples für jeden Kanal
- Packet: kleinste zusammenhängende Datenmenge des Formats

Digitale Audio Verarbeitung

- Interleaved: ein Buffer für beide Kanäle
- Non-Interleaved: ein Buffer pro Kanal

Audio in iOS

- HTML 5 <audio> Tag
- Media Player Framework
- AVFoundation
- OpenAL
- Audio Queues
- Audio Units

- *“Easy” and “CoreAudio” can’t be used in the same sentence.
CoreAudio is very powerful, very complex and under-documented. –
Jens Alfke auf der coreaudio-api Mailling-Liste*

AVAudioSession

- Kategorisiert Apps und regelt den Hardware Zugriff
- Verhalten bzgl. Background Audio
- Verarbeitung von Unterbrechungen
- Reaktion auf Änderungen im Routing
- Für den Vortrag: Kategorie “Playback”

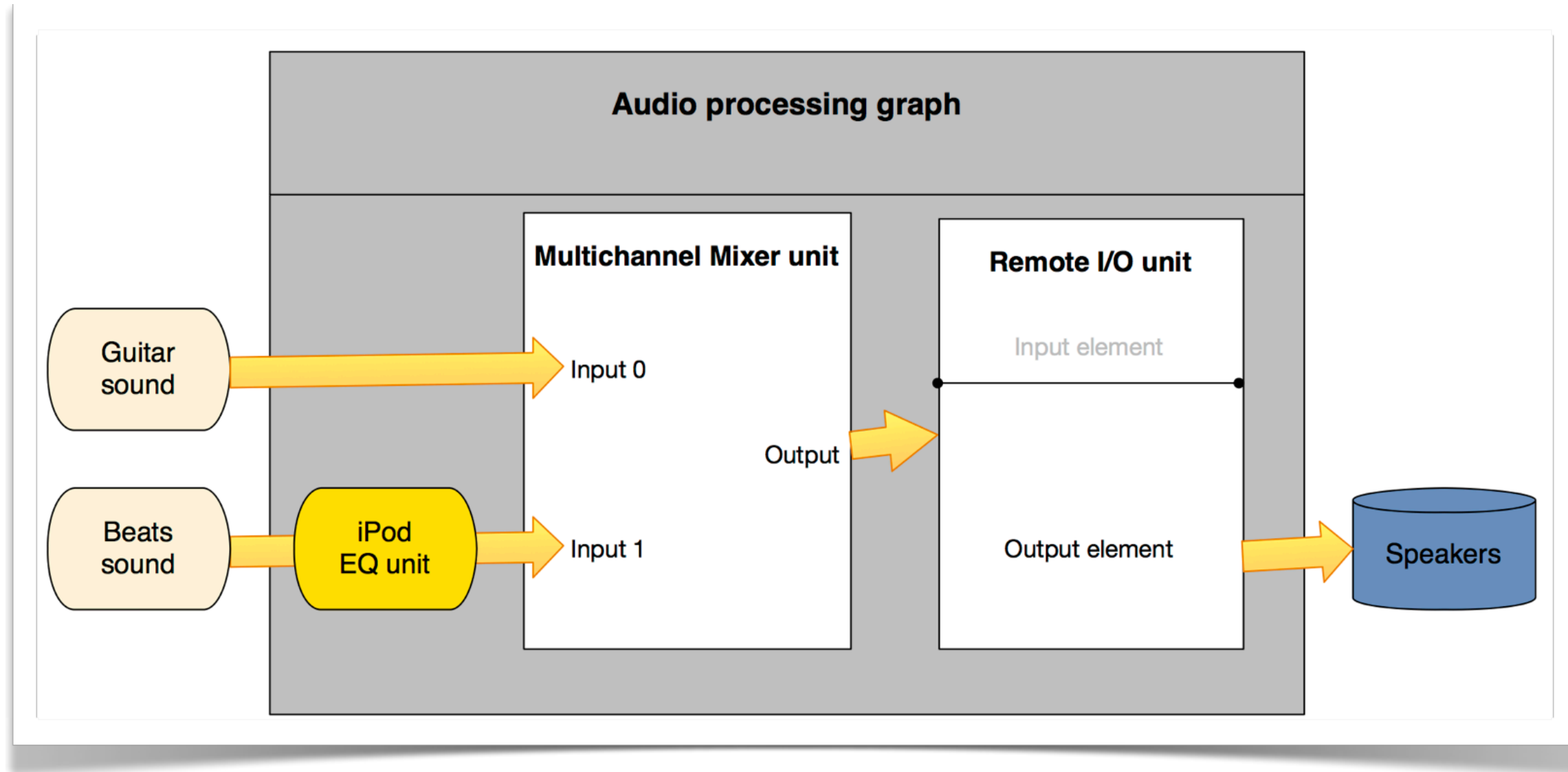
Audio Units

- Echtzeit, Mixing, Audio-Synthese, Streaming
- Größter Sprung in iOS 5
- Reines C
- Callback-lastig, Property-basiert
- Error codes (FourCC oder negative Integers)

Audio Units

- aber auch: niedrigster Level, der Entwicklern auf iOS zur Verfügung steht

Audio Units



AudioUnit

- Audio-Processing Plug-Ins
- iOS: Keine eigenen Units (im Gegensatz zum Mac)
- Mehrere Ein- und Ausgabe-Busse sind möglich
- Verbindungen regeln den Datenfluss

AudioUnit

- *AudioComponentDescription* - Manufacturer, Type, SubType
- *AudioStreamBasicDescription* (ASBD) regelt das Format
- Konfiguration via *AudioUnitSetProperty*
- Dadurch ist die API Doku relativ nutzlos

Verfügbare Audio Units in iOS 4

- *RemotelO, GenericOutput*
- *VoiceProcessingIO*
- *MultiChannelMixer*
- *3DMixer*
- *AUiPodEQ*
- Format Conversion

AUGraph

- Organisiert die Verbindungen zwischen den Units (*AUNode*)
- Hat genau einen Output Knoten
- Arbeitet nach dem Pull-Prinzip
- Mittels *GenericOutput* und Render Callbacks sind Subgraphen möglich

AUGraph

- Drei Phasen: Open, Init, Running
- Änderungen der Sample-Rate nur mittels Neuaufbau und -start
- Bei Änderungen Neustart, sonst kommt es zu Crashes

AURenderCallback

```
OSStatus callback (void* inRefCon,  
                  AudioUnitRenderActionFlags* ioActionFlags,  
                  const AudioTimeStamp* inTimeStamp,  
                  UInt32 inBusNumber,  
                  UInt32 inNumberFrames,  
                  AudioBufferList* ioData);
```

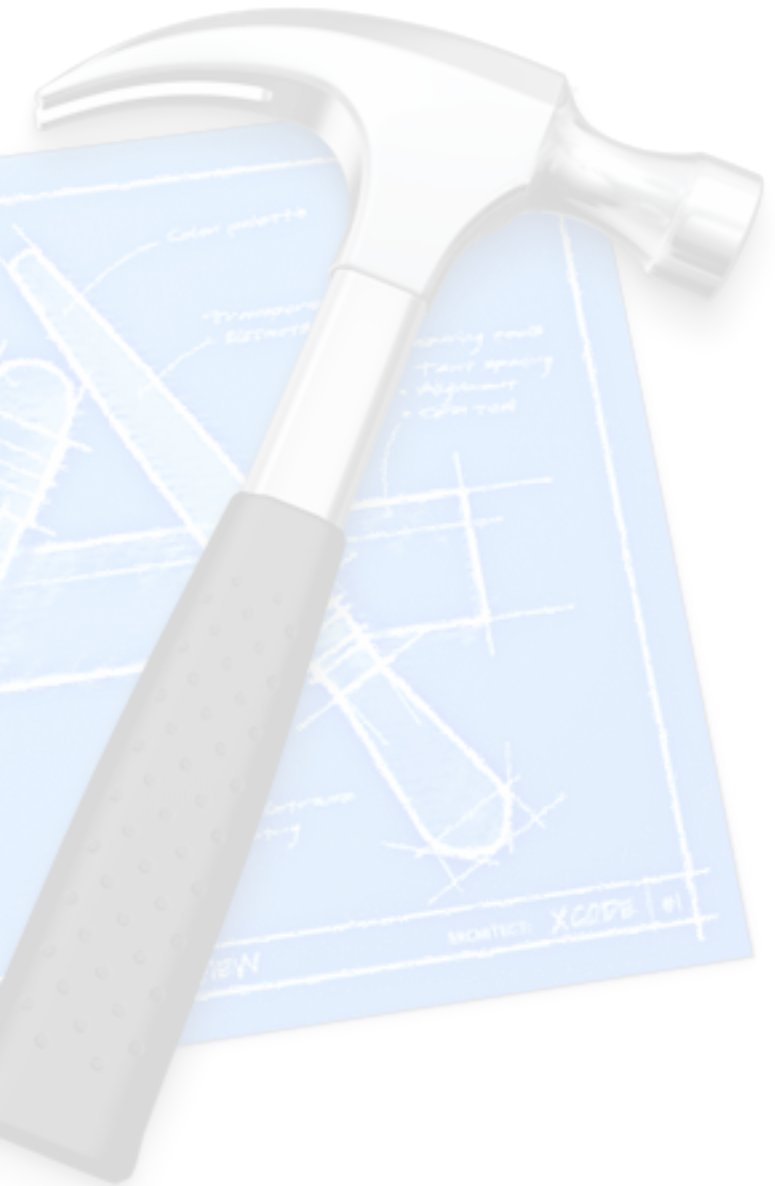
- Hier findet die interessante Arbeit statt!

Erstes Beispiel

Samples abspielen

- *ExtAudioFileOpenUrl()* - unterstützt WAV, AAC, MP3, etc.
- AudioUnits: *MultiChannelMixer* und *Remotelo*
- Ein Render Callback liefert die Samples

Demo



Tipps

Vom AudioTimeStamp zu Nanosekunden

- *AudioTimeStamp* ist in “Host Time”

- Umrechnung:

```
mach_timebase_info_data_t tinfo;  
mach_timebase_info(&tinfo);  
double hTime2nsFactor = (double)  
tinfo.numer / tinfo.denom;  
double nanoseconds = inTimeStamp->mHostTime  
* hTime2nsFactor;
```

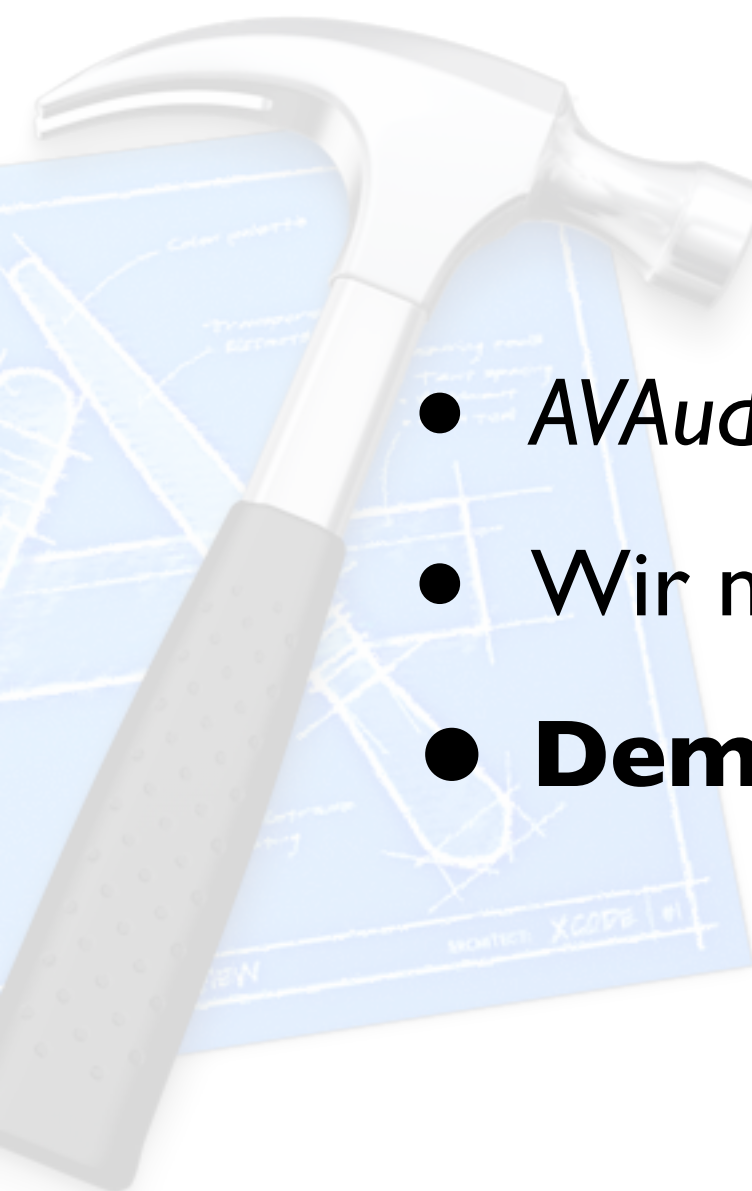

Behandlung von OSStatus

- Jede Funktion gibt einen *OSStatus* zurück
- Dieser muss jedes Mal gecheckt werden!
- Kann FourCC oder Integer Fehlercode sein
- Makro oder Helper-Funktion einsetzen
- Übersichtlicherer Code

Connections vs. AURenderCallback

- Pull-Prinzip aus `_einer_` Quelle
- Daher entweder `Connections` `_oder_` `Callbacks`
- Sonst Error -50 oder `Callbacks` werden nicht aufgerufen
- `CAShow()` hilft beim Debuggen

Level-Meter

- 
- *AVAudioPlayer* hat *peekPowerForChannel*:
 - Wir nutzen einen Render Notifier!
 - **Demo**

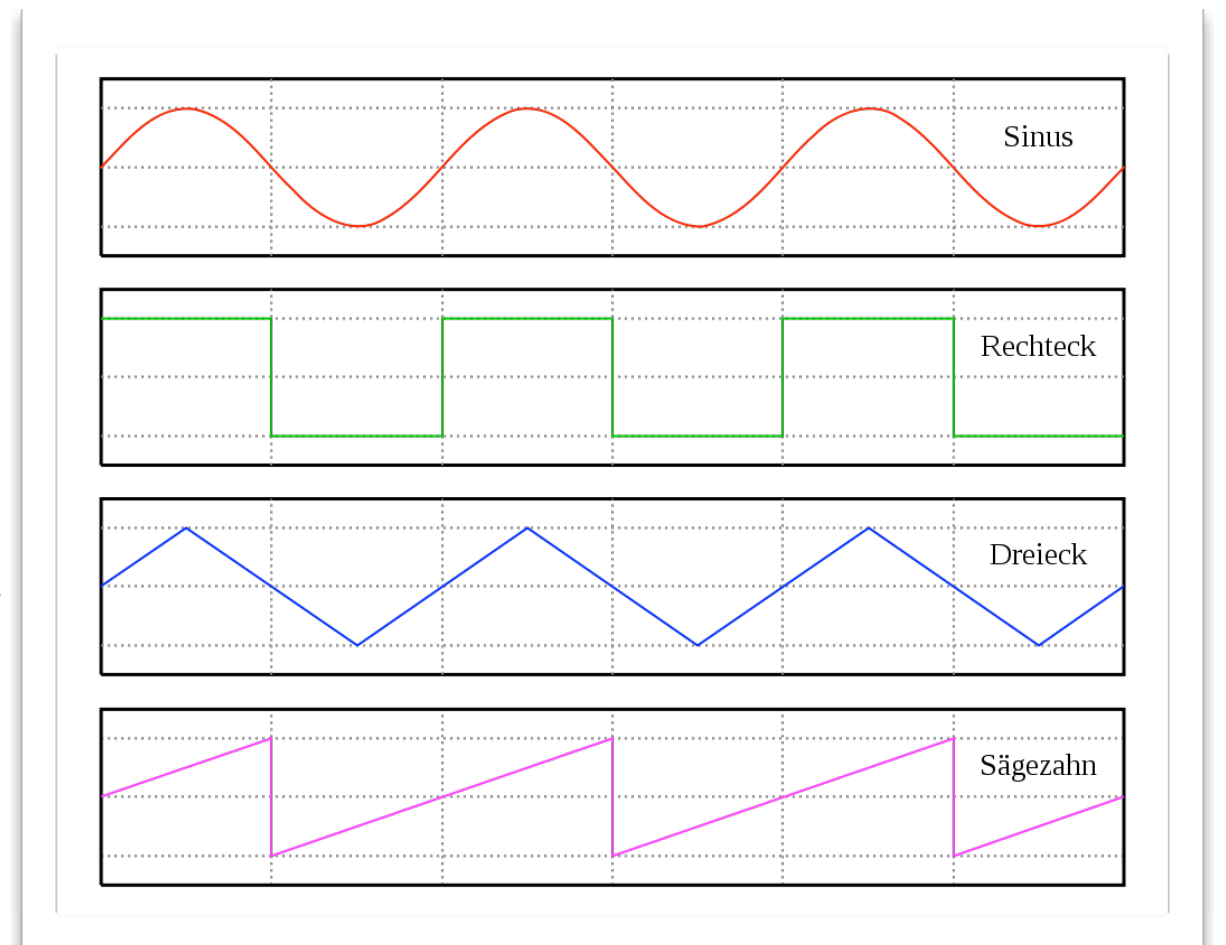
Tipps

- Kein *malloc()* oder *NSLog()* im Render Callback (~85 Aufrufe pro Sekunde!)
- Aber: Zugriff auf Objective-C Objekte via Properties meist OK
- Bei der Arbeit mit ASBD: `memset(&asbd, 0, sizeof(asbd));`
- Umrechnung von Float nach 32bit Signed Int: Multiplikation mit 16777216L

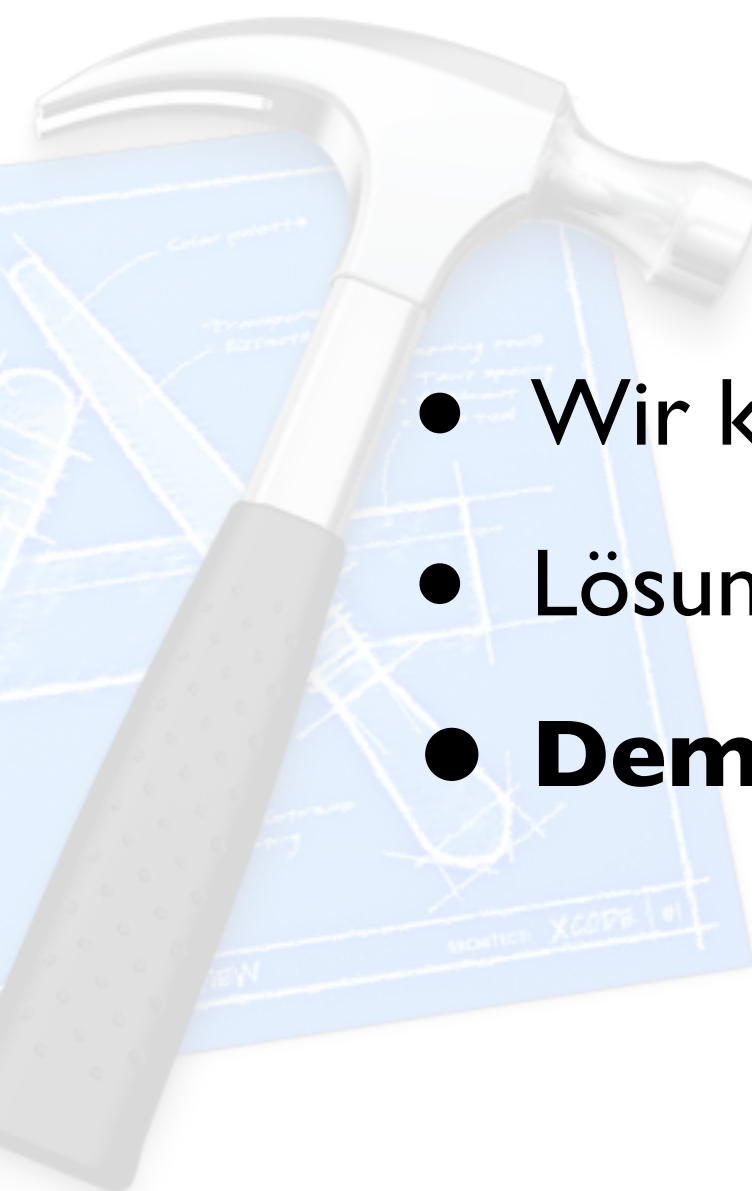
“Kochbuch”

Ton-Erzeugung

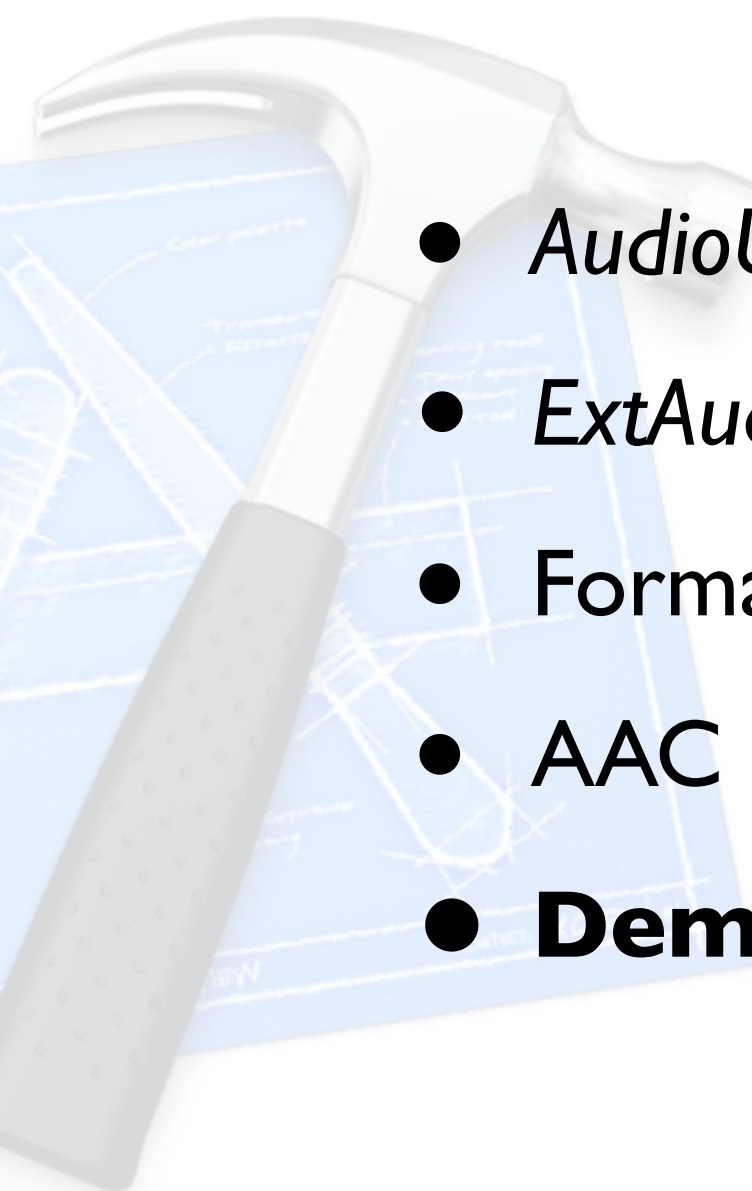
- Elementare Signale
- Erzeugung eines einfachen Sinus-Signals
- **Demo**




Effekt-Filter

- 
- Wir können keine eigenen Audio-Units erzeugen :(
 - Lösung: ausgeschalteter *iPodEQ* mit Render Callback
 - **Demo**

Recording in eine Datei

- 
- *AudioUnitRenderNotify()* auf dem Mixer
 - *ExtAudioFileWriteAsync()*
 - Format beachten!
 - AAC Konvertierung läuft in Hardware
 - **Demo**

Exkurs: SoundFonts

- 
- Samples und Einstellungen für Instrumente in MIDI-Software
 - Freie SoundFonts im Netz, aber kein nativer Support auf iOS
 - Die freie Fluidsynth Library steht unter LGPL
 - Daher: SoundFonts auf dem Mac zu CAF-Dateien verarbeiten
 - **Demo**

Neuerungen in iOS 5

Neue Audio Units

- Effekte: *Filter, Reverb, etc.*
- Generatoren: *AudioFilePlayer, ScheduledSlicePlayer*
- Instrumente: *AUSampler*

AUSampler

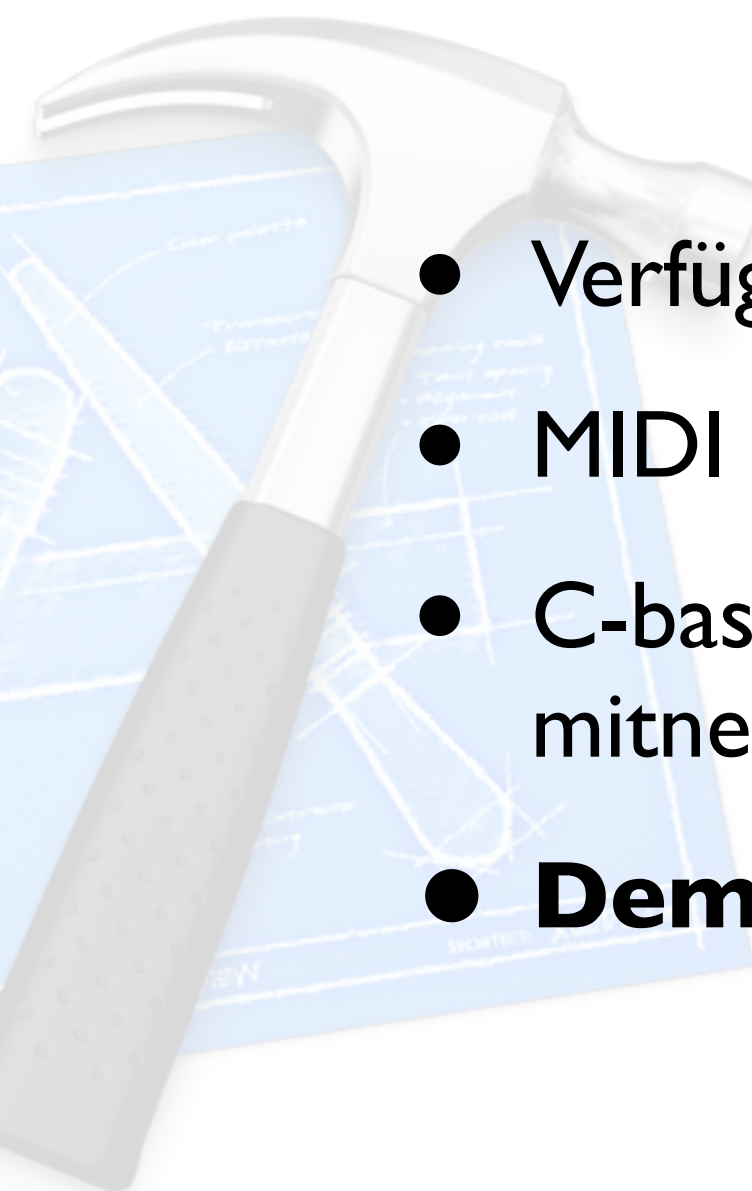
- Neu in iOS 5 und Mac OS X Lion
- Vereinfacht unser Beispiel deutlich
- Native Unterstützung für SoundFonts

Music Sequencing API

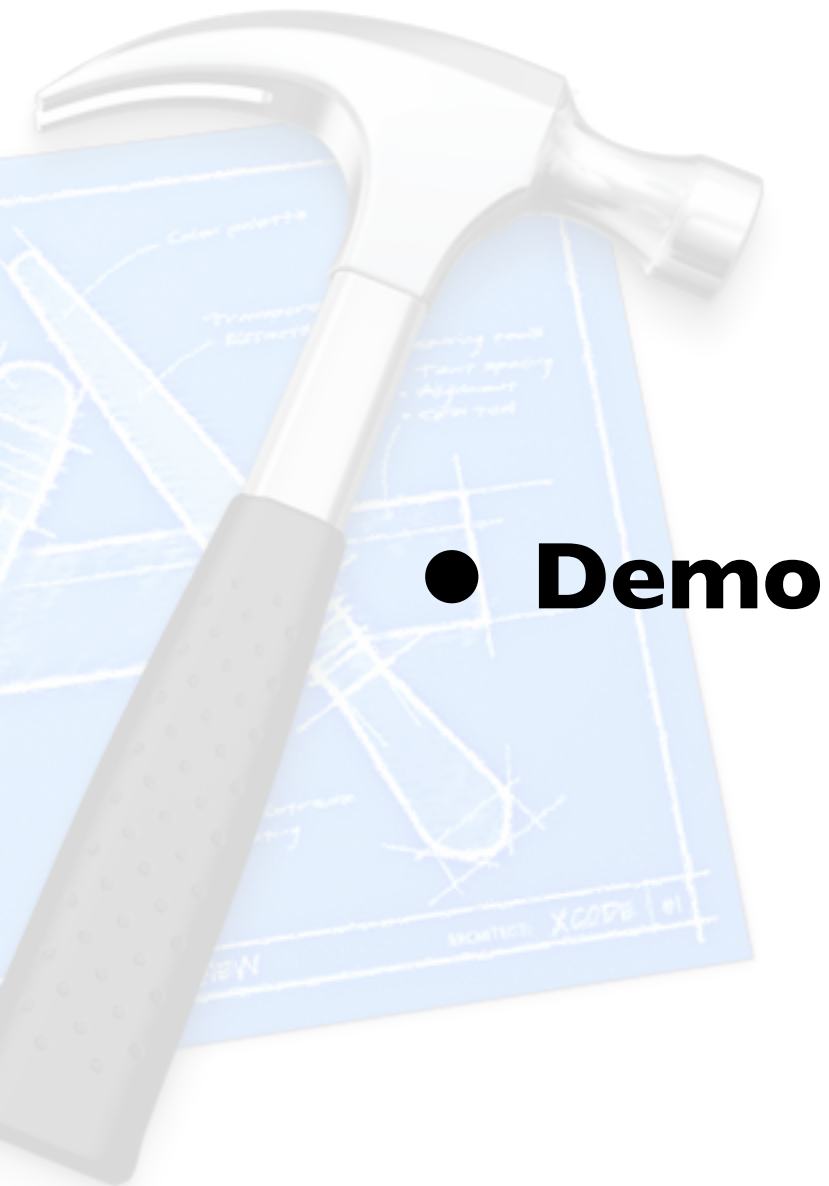
- Lesen und Schreiben von MIDI Files
- Takt
- MusicTracks

Der Synthesizer

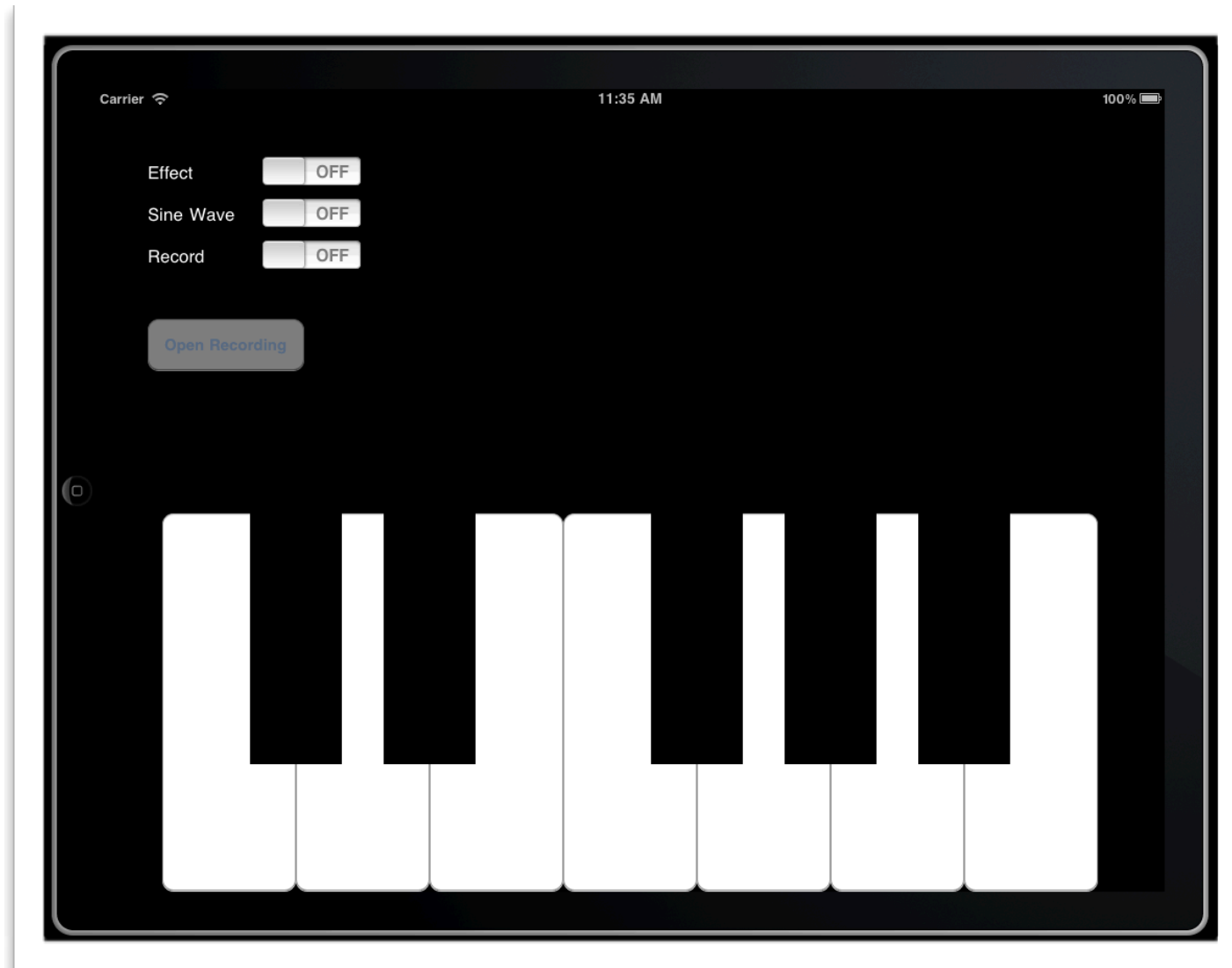
Core MIDI

- 
- Verfügbar seit iOS 4.2
 - MIDI zu USB zu Camera Connection Kit zu iPad
 - C-basiert, wir können unser Wissen von den Audio Units mitnehmen
 - **Demo**

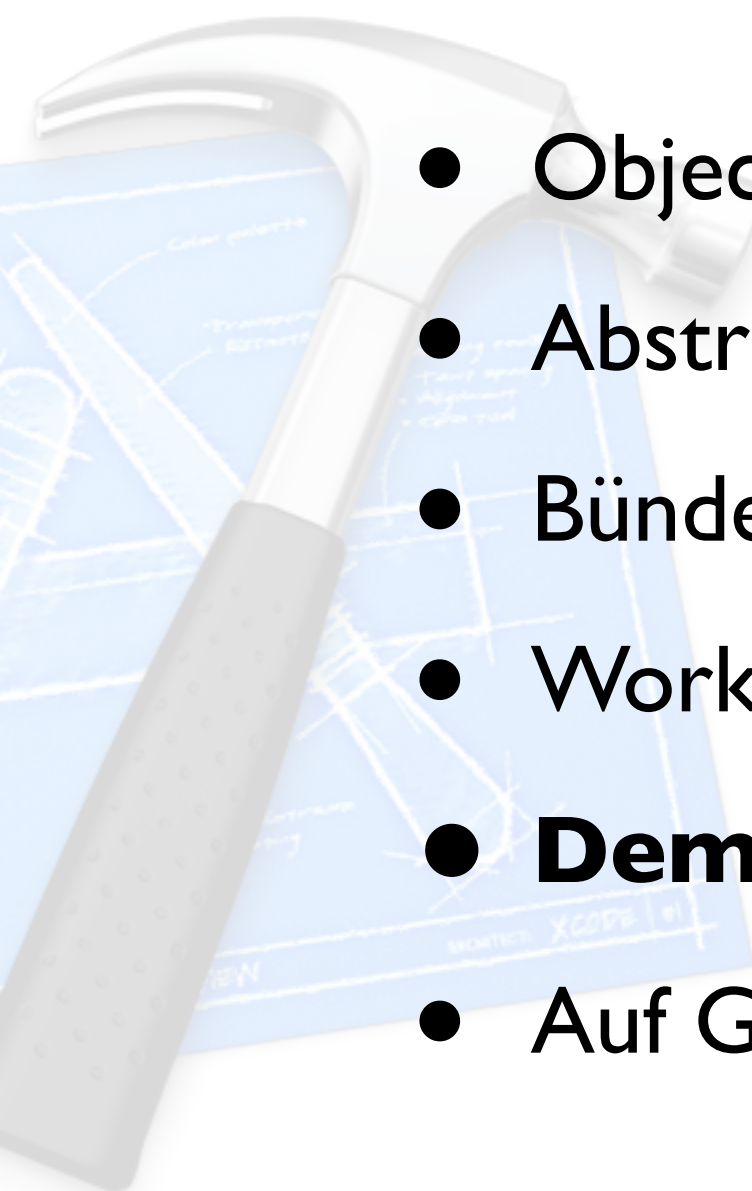
Und alle: der Synthesizer



● Demo



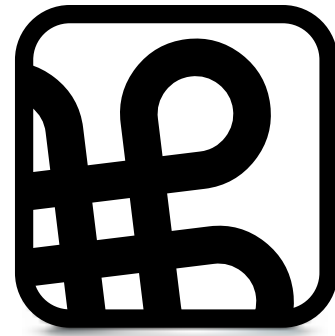
Exkurs: NNAudio

- 
- Objective-C Wrapper um einen Teil der Audio Units
 - Abstrahiert ewig gleiche Setup Blöcke
 - Bündelt *AudioUnit* und *AUNode* in einer Klasse
 - Work-In-Progress
 - **Demo**
 - Auf GitHub: <https://github.com/neonichu/Core-Audio-Samples>

Fragen?

Literatur

- Folien und Beispielcode: <http://vu0.org/audio>
- [Audio Unit Hosting Guide](#) von Apple
- [Core Audio](#) von Chris Adamson und Kevin Avila (Anfang 2012)
- [Fundamentals of Digital Audio](#), WWDC 2010
- [Audio Development for iPhone OS](#), WWDC 2010
- [Music in iOS and Lion](#), WWDC 2011



Macoun' I I