Audio Unit Extensions

Session 508

Doug Wyatt Core Audio Plumber

Audio Units

Since OS X 10.0, iOS 2.0

OS includes many units

- I/O, mixers, effects, more
- Used in higher-level API's (e.g. media playback)

3rd-party plug-ins (OS X)

Audio Unit Extensions

NEW

Full plug-in model

OS X and iOS

App Extensions -> App Store!

New API

- Modern
- Compatible

Version 3 Audio Unit API

v1: 2001

v2: 2002

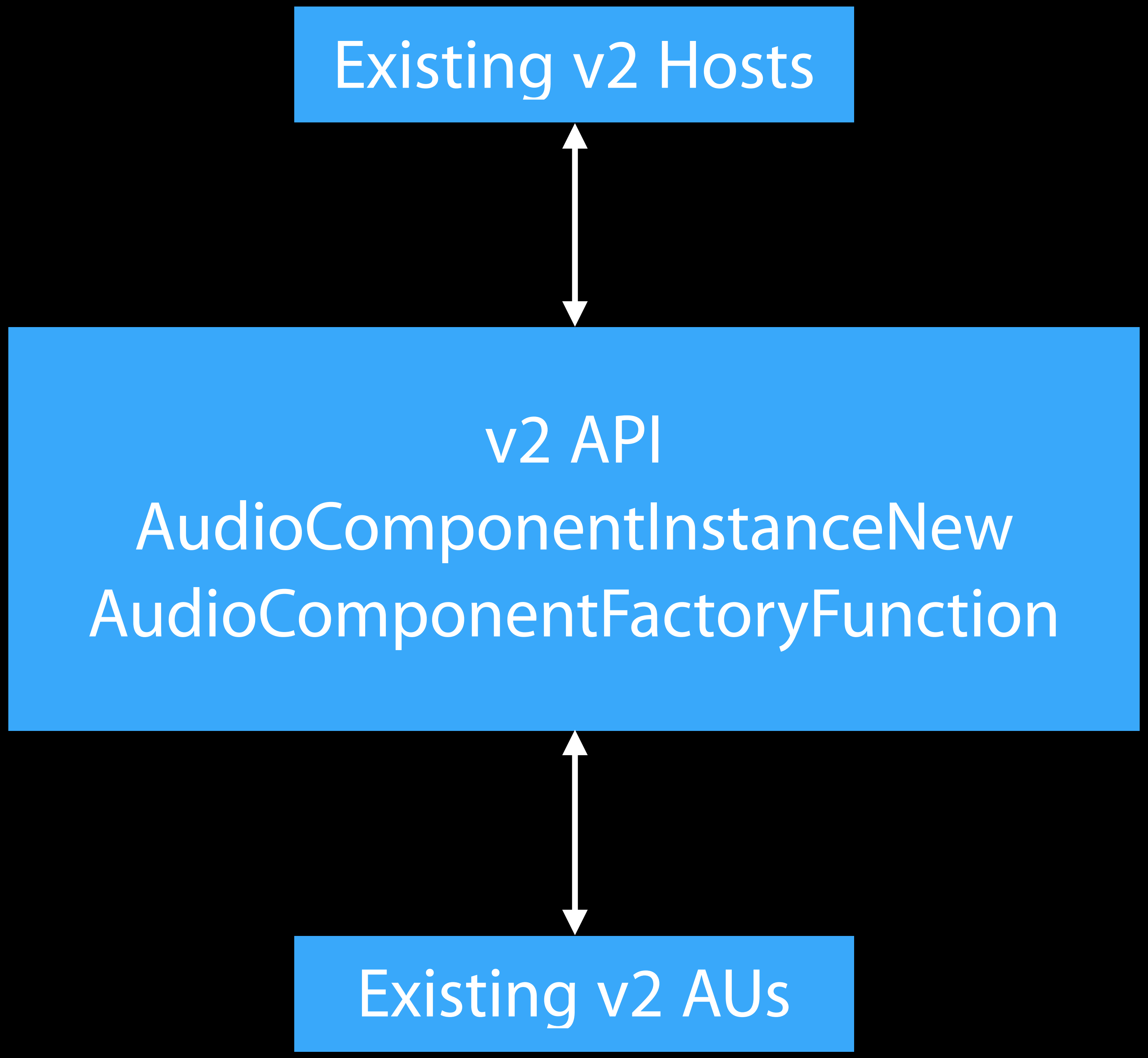
v3: AUAudioUnit class

- In AudioUnit.framework
- Objective-C / Swift

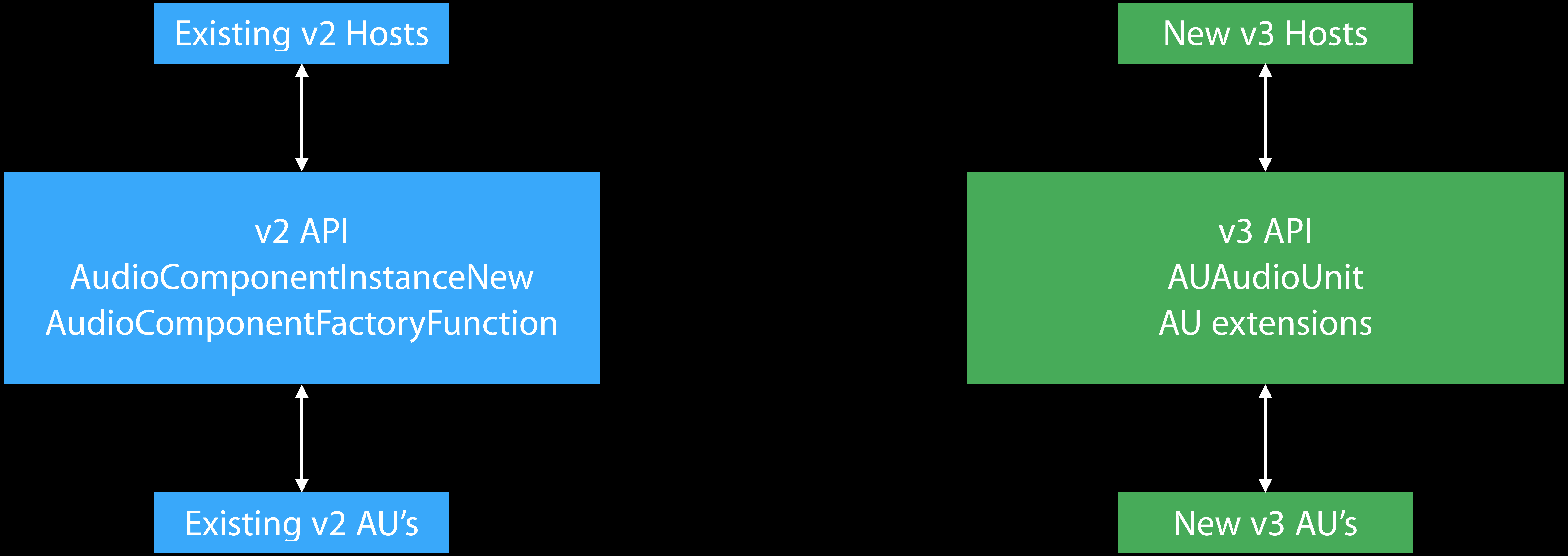
AVFoundation

AVAudioUnitComponentManager	Locate Audio Unit components	OS X 10.10+ iOS 9.0+
AVAudioUnitComponent	An Audio Unit component	OS X 10.10+ iOS 9.0+
AVAudioUnit AVAudioUnitEffect etc.	Audio Unit instance, in AVAudioEngine	OS X 10.10+ iOS 8.0+

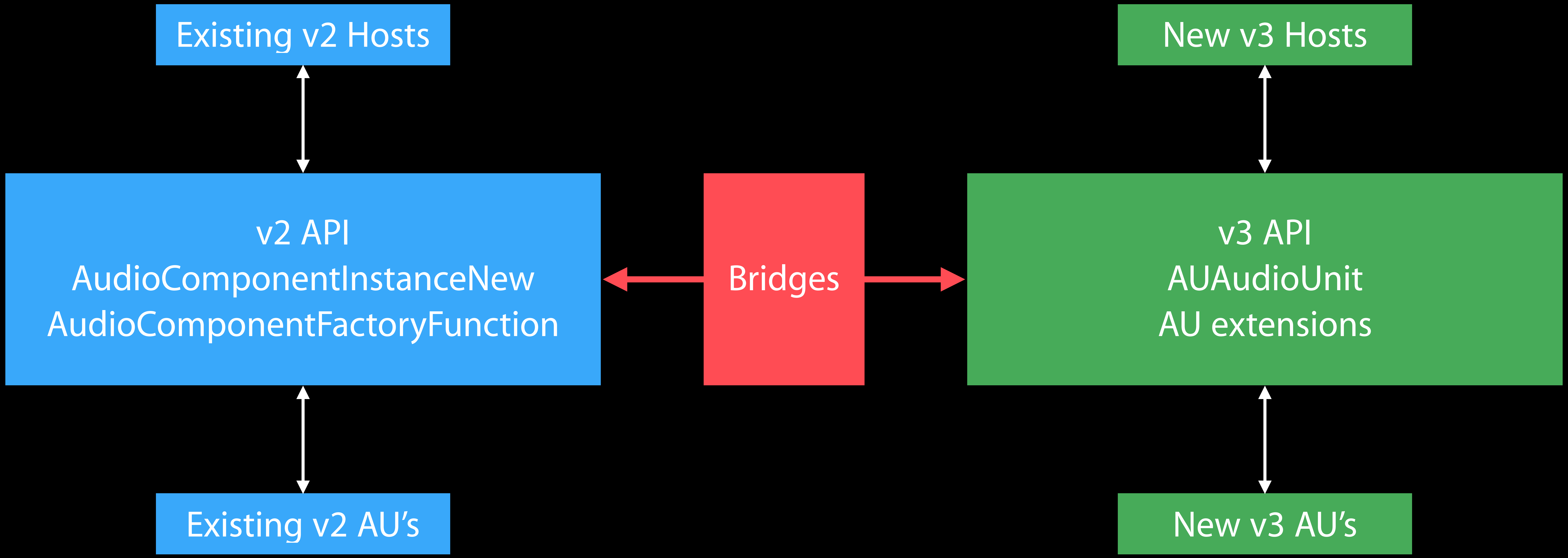
Compatibility



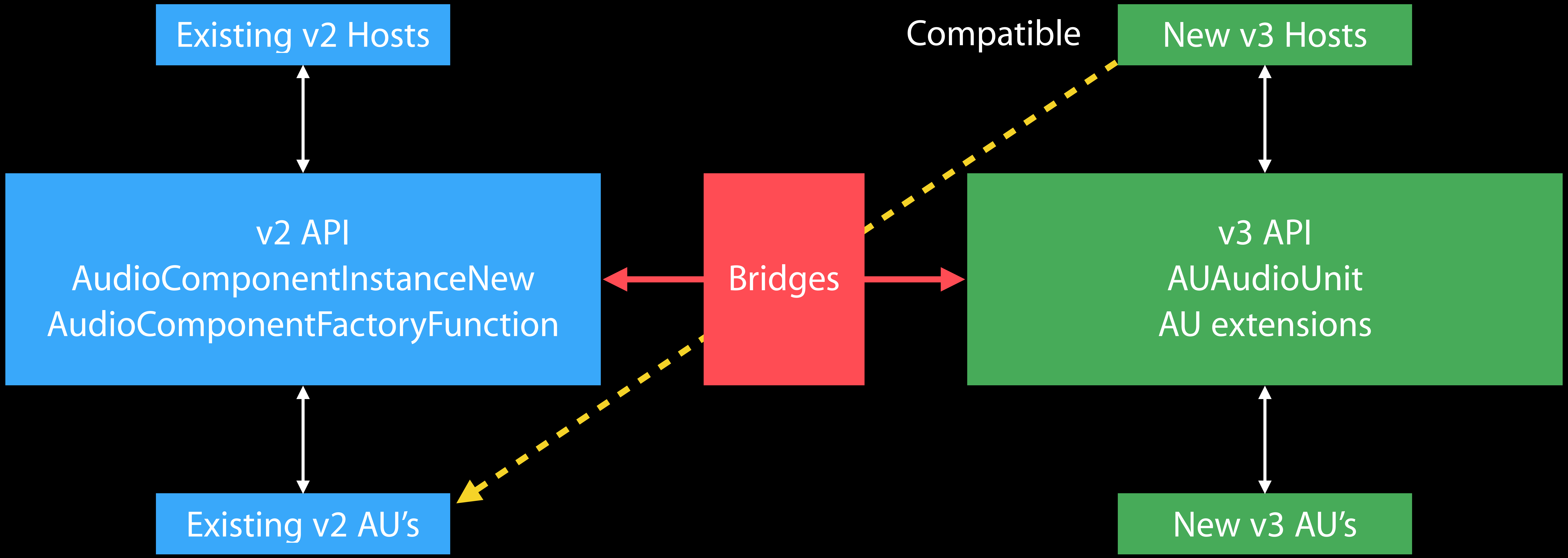
Compatibility



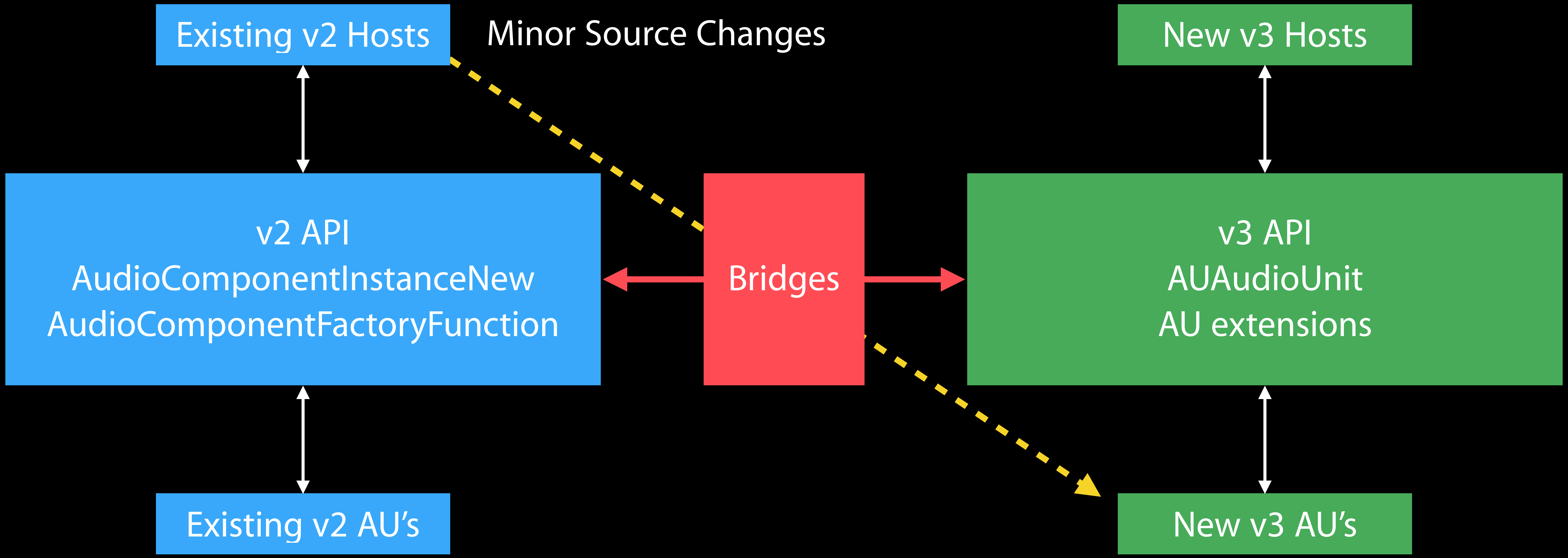
Compatibility



Compatibility



Compatibility



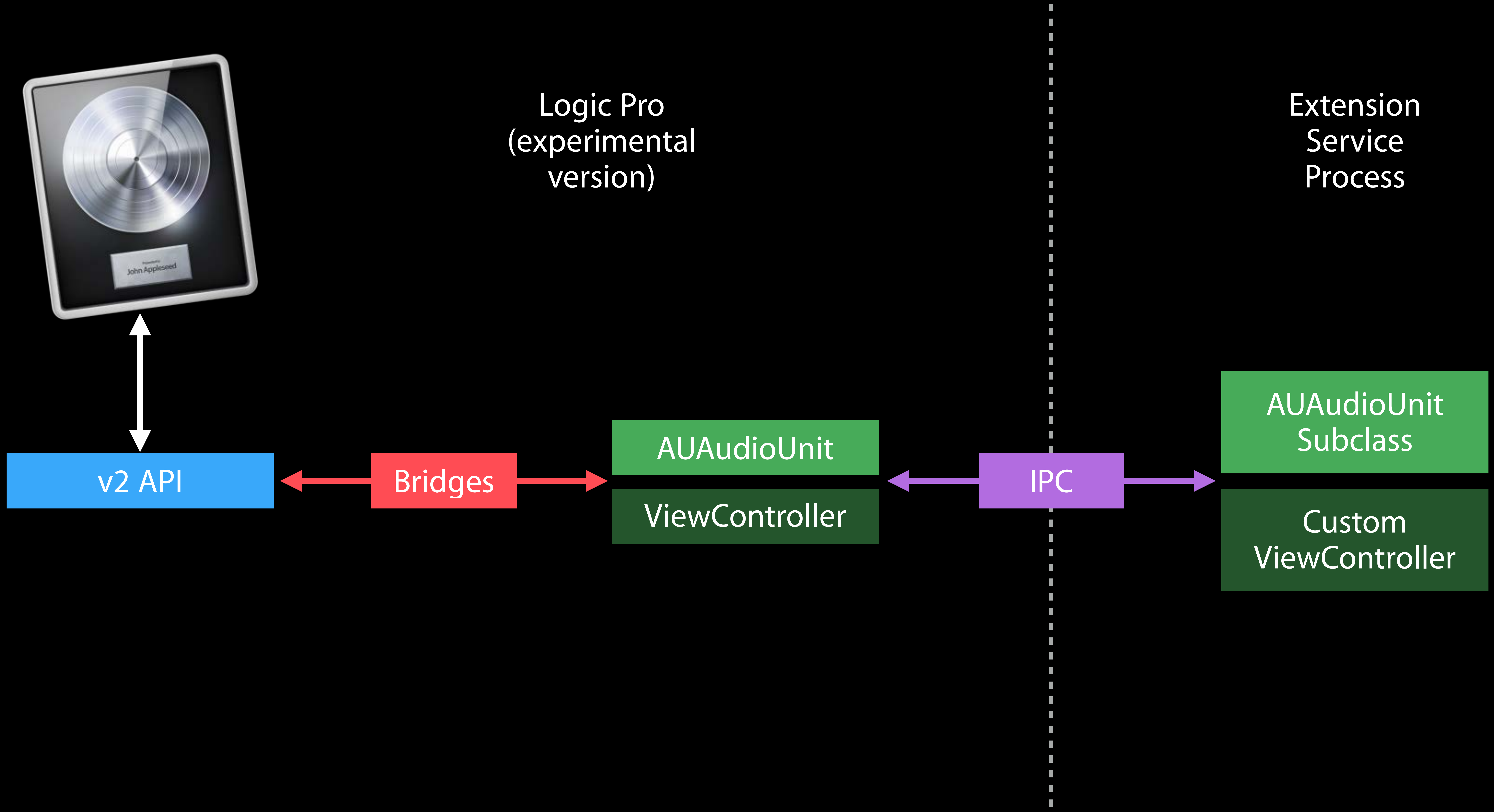
Demo

Audio Unit Extension in Logic Pro

Doug Wyatt

Core Audio Plumber

Logic Pro with AU Extension



Hosting Audio Units

Hosting Audio Units

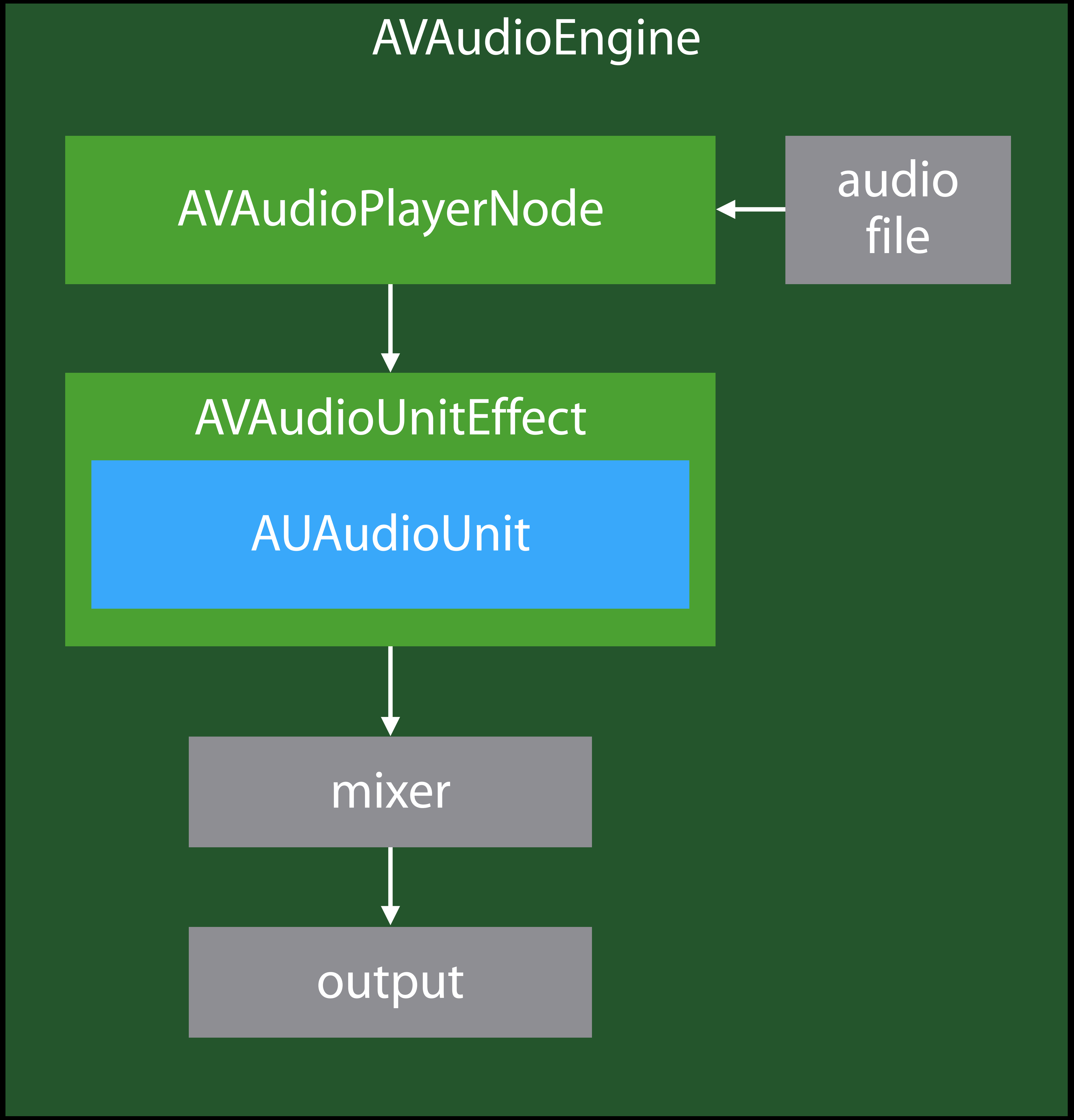
Using v3 API's

Sample Code: `AudioUnitV3Example`

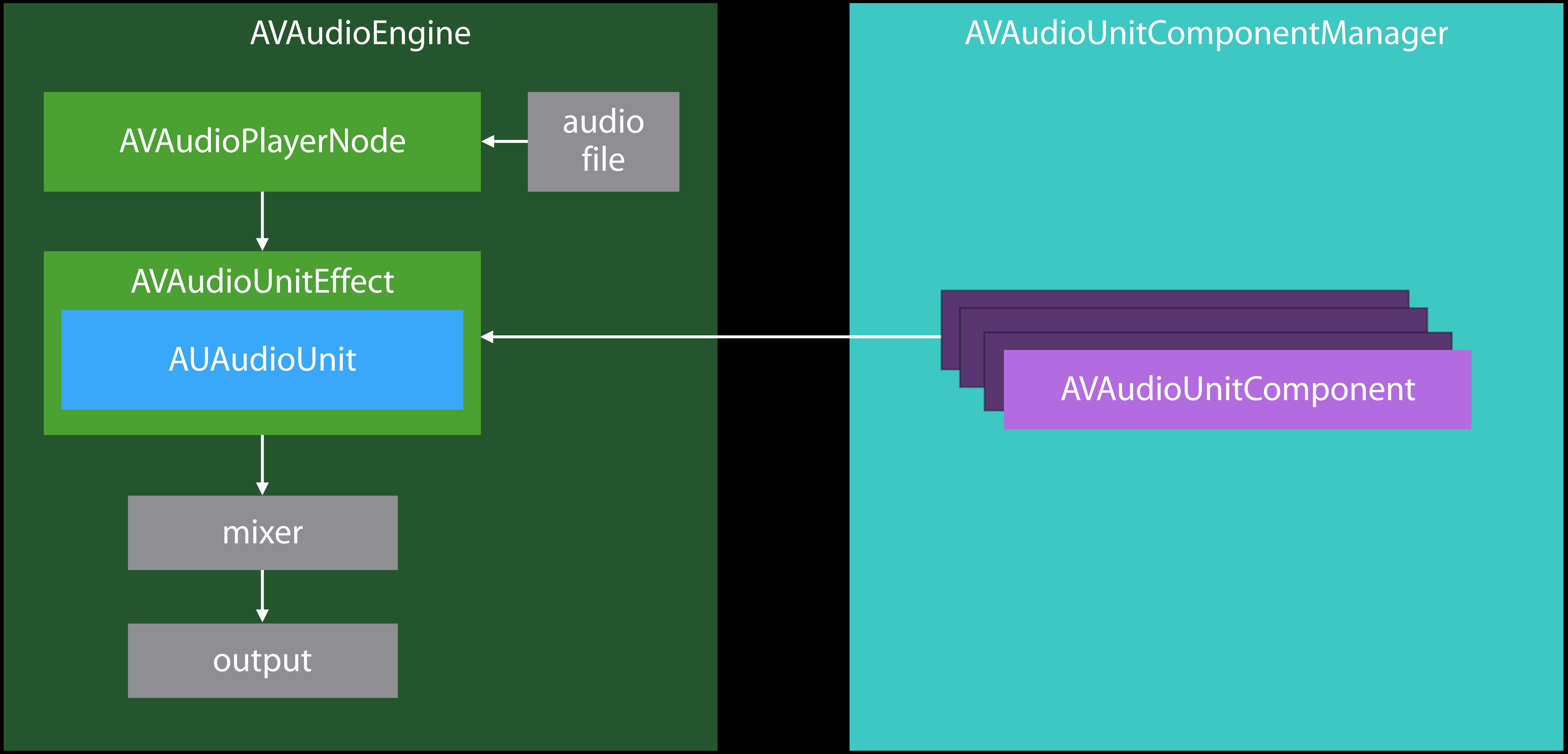
AUHost shows

- Finding
- Opening
- Connecting
- Presets
- Opening AU's custom view

SimplePlayEngine



SimplePlayEngine



Audio Components

```
struct AudioComponentDescription {
  var componentType: OSType
  var componentSubType: OSType
  var componentManufacturer: OSType
  var componentFlags: UInt32
  var componentFlagsMask: UInt32
}
```

Finding Audio Unit Components

```
let anyEffect = AudioComponentDescription(componentType:
kAudioUnitType_Effect, componentSubType: 0, componentManufacturer:
0, componentFlags: 0, componentFlagsMask: 0)

let availableEffects: [AVAudioUnitComponent] =
AVAudioUnitComponentManager.sharedAudioUnitComponentManager().
componentsMatchingDescription(anyEffect)
```

Finding Audio Unit Components

```
let anyEffect = AudioComponentDescription(componentType:
kAudioUnitType_Effect, componentSubType: 0, componentManufacturer:
0, componentFlags: 0, componentFlagsMask: 0)
```

```
let availableEffects: [AVAudioUnitComponent] =
AVAudioUnitComponentManager.sharedAudioUnitComponentManager().
componentsMatchingDescription(anyEffect)
```

Finding Audio Unit Components

```
let anyEffect = AudioComponentDescription(componentType:
kAudioUnitType_Effect, componentSubType: 0, componentManufacturer:
0, componentFlags: 0, componentFlagsMask: 0)

let availableEffects: [AVAudioUnitComponent] =
AVAudioUnitComponentManager.sharedAudioUnitComponentManager().
componentsMatchingDescription(anyEffect)
```

Finding Audio Unit Components

```
let anyEffect = AudioComponentDescription(componentType:
kAudioUnitType_Effect, componentSubType: 0, componentManufacturer:
0, componentFlags: 0, componentFlagsMask: 0)
```

```
let availableEffects: [AVAudioUnitComponent] =
AVAudioUnitComponentManager.sharedAudioUnitComponentManager().
componentsMatchingDescription(anyEffect)
```

Selecting an Audio Unit

```
func selectEffectComponent(comp: AVAudioUnitComponent?,
completionHandler: () -> Void) {
  selectEffectWithComponentDescription
    (comp?.audioComponentDescription,
    completionHandler: completionHandler)
}
```

Creating an AudioUnit Instance

With AVAudioEngine

```
AVAudioUnit.instantiateWithComponentDescription(desc!, options:
[]) {
    avAudioUnit, error in
    guard let avAudioUnit = avAudioUnit else { return }
    self.effect = avAudioUnit
    self.engine.attachNode(self.effect!)
    …
```

Creating an AudioUnit Instance

With AVAudioEngine

```
AVAudioUnit.instantiateWithComponentDescription(desc!, options:
[]) {
    avAudioUnit, error in
    guard let avAudioUnit = avAudioUnit else { return }
   self.effect = avAudioUnit
    self.engine.attachNode(self.effect!)
    …
```

Creating an AudioUnit Instance

With AVAudioEngine

```
AVAudioUnit.instantiateWithComponentDescription(desc!, options:
[]) {
    avAudioUnit, error in

    guard let avAudioUnit = avAudioUnit else { return }

    self.effect = avAudioUnit
    self.engine.attachNode(self.effect!)
    …
```

Creating an AudioUnit Instance

With AVAudioEngine

```
AVAudioUnit.instantiateWithComponentDescription(desc!, options:
[]) {
    avAudioUnit, error in

    guard let avAudioUnit = avAudioUnit else { return }

    self.effect = avAudioUnit
    self.engine.attachNode(self.effect!)
    …
```

Creating an AudioUnit Instance

With AVAudioEngine

```
AVAudioUnit.instantiateWithComponentDescription(desc!, options:
[]) {
    avAudioUnit, error in

    guard let avAudioUnit = avAudioUnit else { return }

    self.effect = avAudioUnit
    self.engine.attachNode(self.effect!)
    …
```

Creating an AudioUnit Instance

With AVAudioEngine

```
AVAudioUnit.instantiateWithComponentDescription(desc!, options:
[]) {
    avAudioUnit, error in

    guard let avAudioUnit = avAudioUnit else { return }

    self.effect = avAudioUnit
    self.engine.attachNode(self.effect!)
    …
```

Creating an AudioUnit Instance

With AVAudioEngine

```
self.engine.disconnectNodeInput(self.engine.mainMixerNode)


self.engine.connect(self.player, to: self.effect!,
  format: self.file!.processingFormat)


self.engine.connect(self.effect!, to: self.engine.mainMixerNode,
  format: self.file!.processingFormat)
```

Creating an AudioUnit Instance

With AVAudioEngine

```
self.engine.disconnectNodeInput(self.engine.mainMixerNode)


self.engine.connect(self.player, to: self.effect!,
  format: self.file!.processingFormat)


self.engine.connect(self.effect!, to: self.engine.mainMixerNode,
  format: self.file!.processingFormat)
```

Creating an AudioUnit Instance

With AVAudioEngine

```
self.engine.disconnectNodeInput(self.engine.mainMixerNode)

self.engine.connect(self.player, to: self.effect!,
  format: self.file!.processingFormat)

self.engine.connect(self.effect!, to: self.engine.mainMixerNode,
  format: self.file!.processingFormat)
```

Creating an AudioUnit Instance

With AVAudioEngine

```
self.engine.disconnectNodeInput(self.engine.mainMixerNode)
```

```
self.engine.connect(self.player, to: self.effect!,
  format: self.file!.processingFormat)
```

```
self.engine.connect(self.effect!, to: self.engine.mainMixerNode,
  format: self.file!.processingFormat)
```

Creating an AudioUnit Instance

With AVAudioEngine

```
self.engine.disconnectNodeInput(self.engine.mainMixerNode)

self.engine.connect(self.player, to: self.effect!,
  format: self.file!.processingFormat)

self.engine.connect(self.effect!, to: self.engine.mainMixerNode,
  format: self.file!.processingFormat)
```

Creating an AudioUnit Instance

With AVAudioEngine

```
self.engine.disconnectNodeInput(self.engine.mainMixerNode)


self.engine.connect(self.player, to: self.effect!,
  format: self.file!.processingFormat)


self.engine.connect(self.effect!, to: self.engine.mainMixerNode,
  format: self.file!.processingFormat)
```

Creating an AudioUnit Instance

With AVAudioEngine

```
self.audioUnit = self.effect!.AUAudioUnit


if let pl = self.audioUnit!.factoryPresets {
  self.presetList = pl
}
else {
  self.presetList = [AUAudioUnitPreset]()
}
```

Creating an AudioUnit Instance

With AVAudioEngine

```
self.audioUnit = self.effect!.AUAudioUnit


if let pl = self.audioUnit!.factoryPresets {
    self.presetList = pl
}
else {
    self.presetList = [AUAudioUnitPreset]()
}
```

Creating an AudioUnit Instance

With AVAudioEngine

```
self.audioUnit = self.effect!.AUAudioUnit

if let pl = self.audioUnit!.factoryPresets {
  self.presetList = pl
}
else {
  self.presetList = [AUAudioUnitPreset]()
}
```

Creating an AudioUnit Instance

With AVAudioEngine

```
self.audioUnit = self.effect!.AUAudioUnit

if let pl = self.audioUnit!.factoryPresets {
  self.presetList = pl
}
else {
  self.presetList = [AUAudioUnitPreset]()
}
```

Accessing an Audio Unit’s View Controller

```
playEngine.audioUnit?.requestViewControllerWithCompletionHandler {
  [weak self] viewController in
 … embed audio unit’s view controller’s view …
}
```

Demo

AUHost

Michael Hopkins
Purveyor of Pixels

Using an AudioUnit

Directly, v3 (Swift):

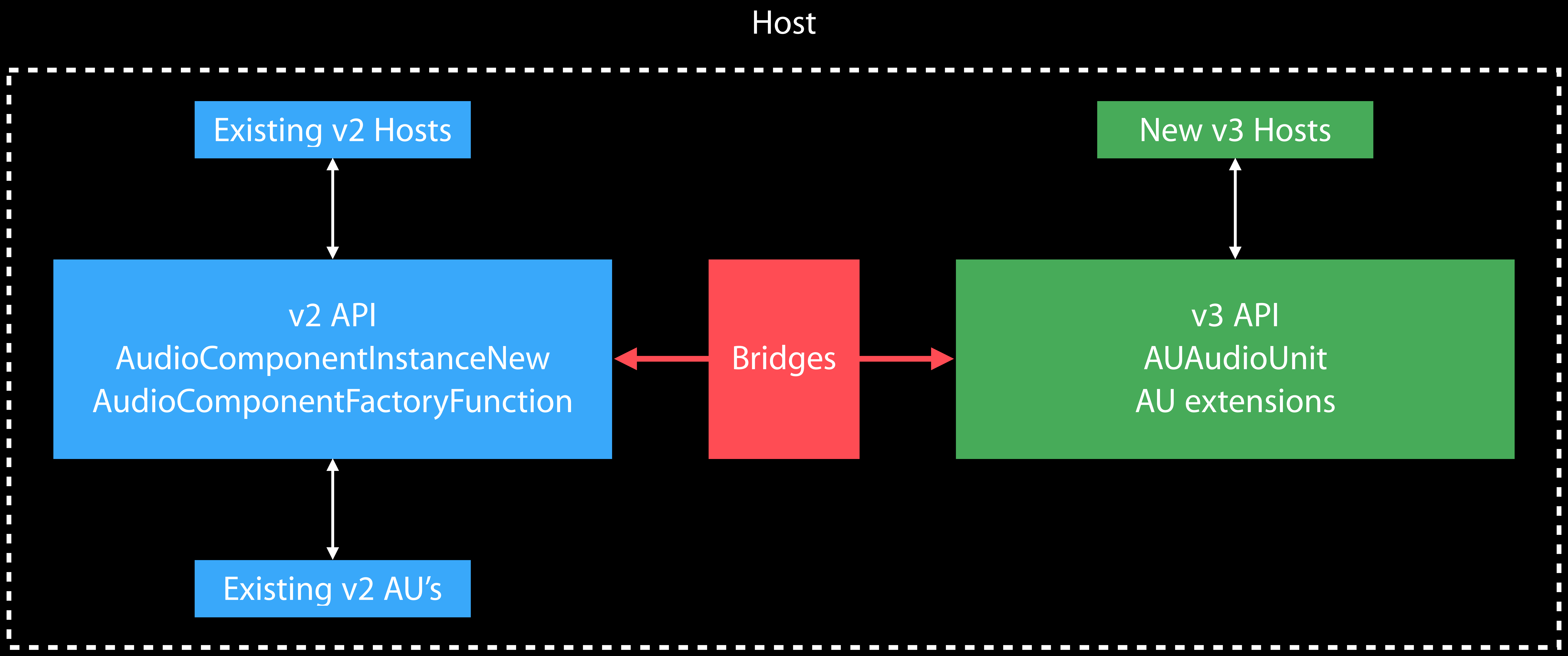
```
AUAudioUnit.instantiateWithComponentDescription(desc, options: [])
{ audioUnit, err in
        …
    }
```

Directly, v2 (C):

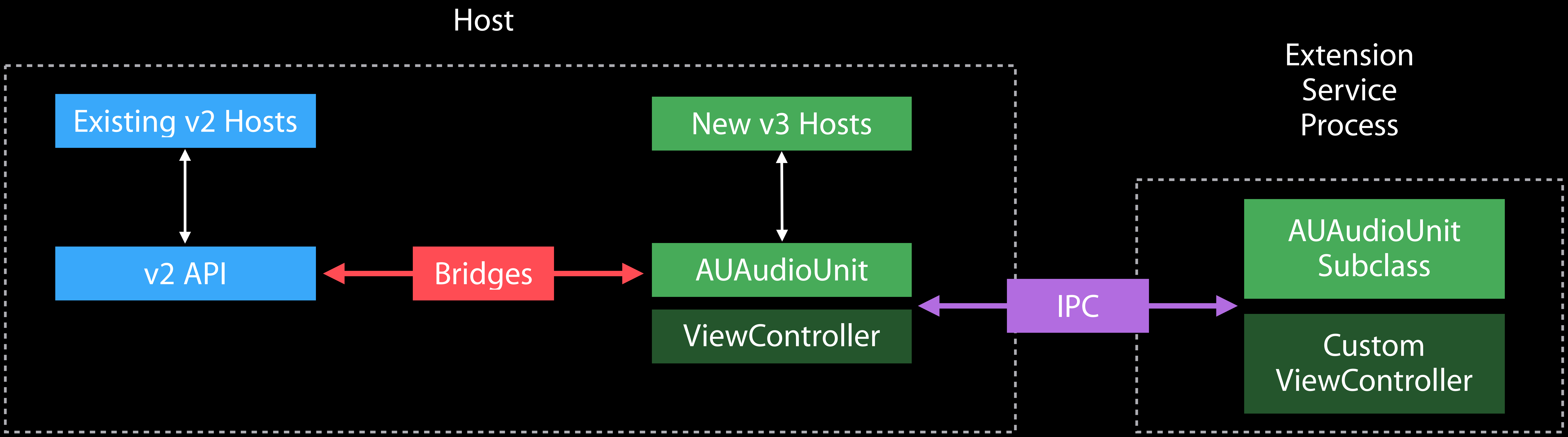
```
AudioComponentInstantiate(desc, options,
    ^(AudioComponentInstance au, OSStatus err) {
        …
    })
```

Hosting: In Versus Out of Process

v2 AU's: Always in Host's Process



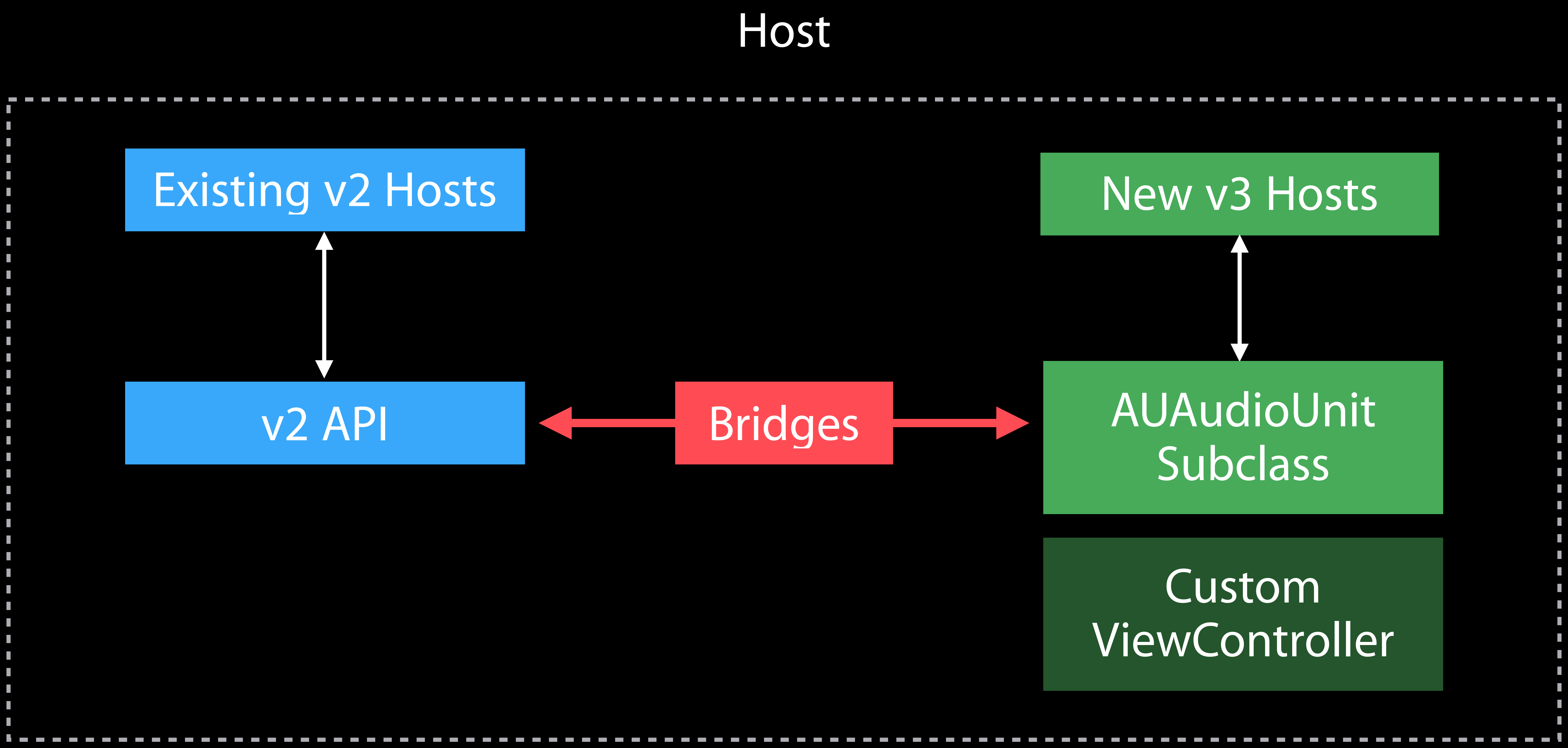
v3 AU's: Default to Separate Process



v3 AU's: Optionally In-Process

Host options: `kAudioComponentInstantiation_LoadInProcess`

AU extension plist entry: `AudioComponentBundle`



In-Process: Safety Versus Performance

Safety

- Security risk
- Misbehaving AU can crash the host

Performance

- IPC overhead ~40 µs per render cycle
- Significance depends on number of AU's and render interval
- 0.1% at 1024 frames / 44.1 kHz
- 5.5% at 32 frames / 44.1 kHz

Updating a v2 Host

If kAudioComponentFlag_RequiresAsyncInstantiation is set, use:

```
extern void AudioComponentInstantiate(
            AudioComponent                    inComponent,
            AudioComponentInstantiationOptions  inOptions,
            void (^inCompletionHandler)(
   AudioComponentInstance __nullable, OSStatus));
```

Instead of:

```
extern OSStatus AudioComponentInstanceNew(AudioComponent
inComponent, AudioComponentInstance *outInstance);
```

Updating a v2 Host

If `kAudioComponentFlag_IsV3AudioUnit` is set, use:

`kAudioUnitProperty_RequestViewController`

(also asynchronous)

Instead of:

`kAudioUnitProperty_CocoaUI`

Asynchronous Operations

Can use new methods with v2 units

Must use new methods with v3 units

Helps responsiveness

- Unblocks main thread

Don’t wait on the main thread!

- Deadlock

Creating an Audio Unit Extension

About App Extensions

.appex

App’s PlugIns directory

Runs in XPC service process

See *App Extension Programming Guide*

Creating an Audio Unit

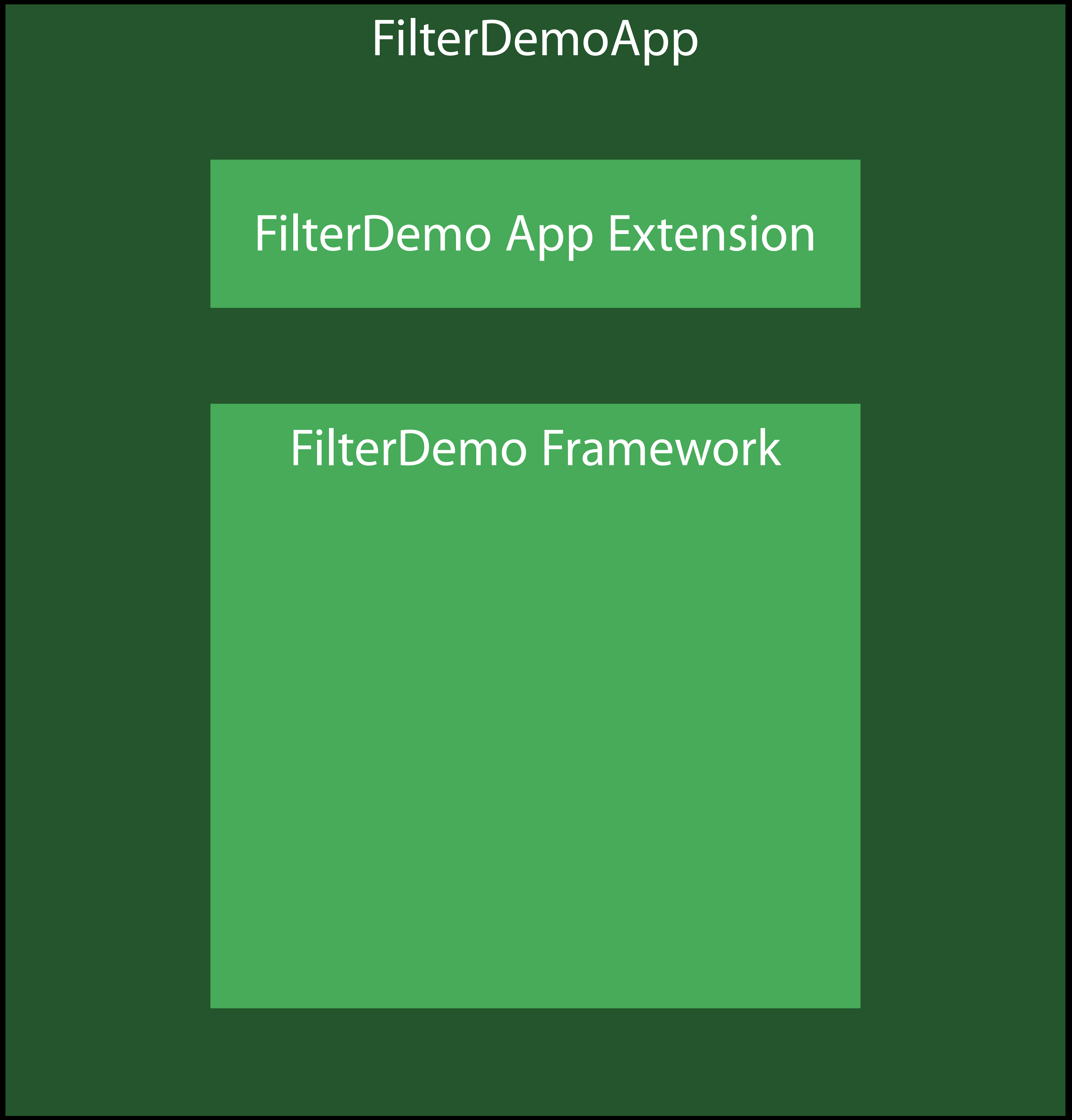
Using v3 API's

New Sample Code: `AudioUnitV3Example`

FilterDemo

FilterDemo Anatomy

- Containing app
- Extension (.appex)
- Both link framework

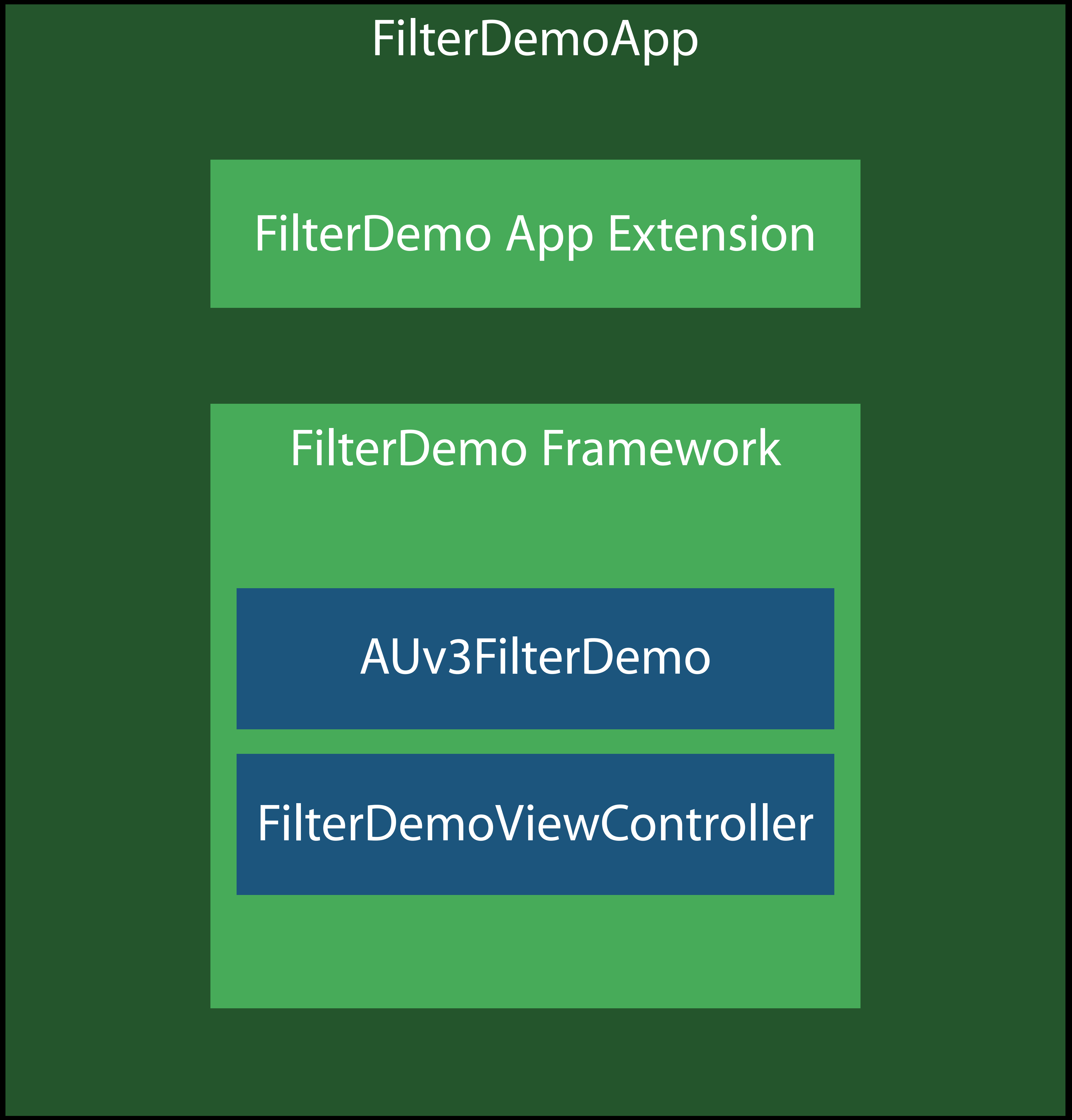


FilterDemo Anatomy

Containing app

Extension

Both link framework



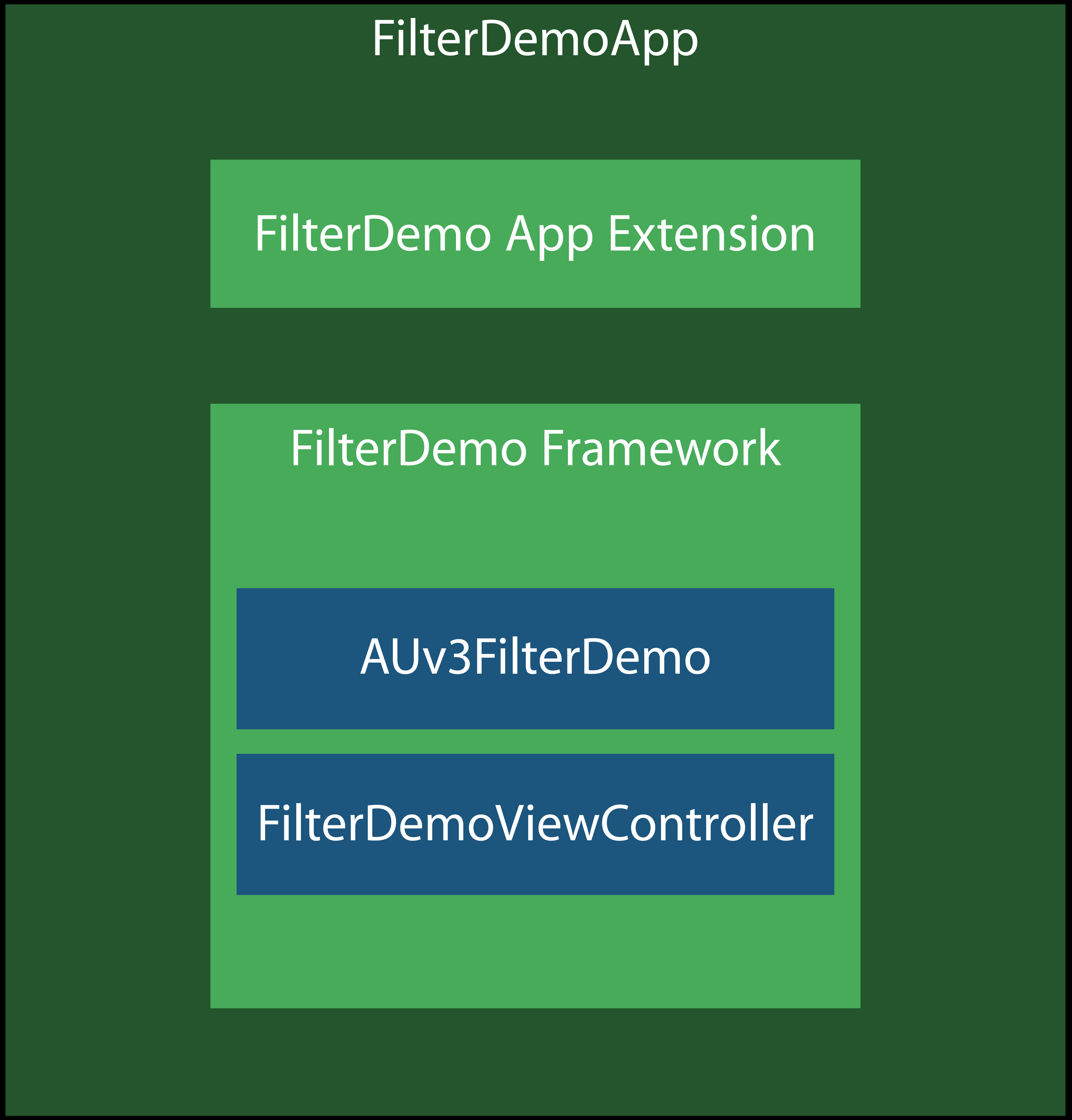
FilterDemo Anatomy

Containing app

Extension

Both link framework

- Easy development within app (no XPC)
- Code size/duplication
- On OS X, framework can be loaded into host process



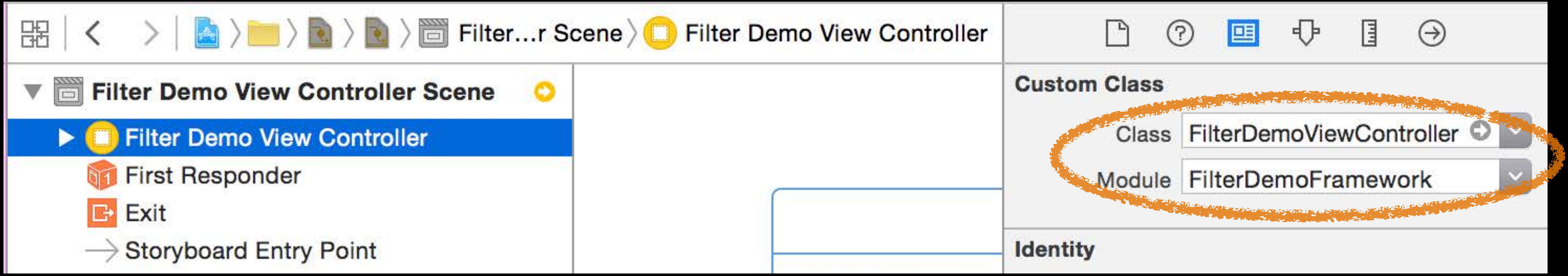
Extension's Info.plist

Comments in <AudioUnit/AUAudioUnitImplementation.h>

NSExtensionPointIdentifier	`com.apple.AudioUnit-UI`
NSExtensionMainStoryboard	MainInterface
AudioComponents	`AudioComponentDescription` (type/subtype/manufacturer), name, version, tags

Extension's MainInterface.storyboard

Entry point is a View Controller, set its Custom Class:


Filter...r Scene
Filter Demo View Controller
Filter Demo View Controller Scene
Filter Demo View Controller
First Responder
Exit
Storyboard Entry Point
Custom Class
Identity

Class FilterDemoViewController

Module FilterDemoFramework

Extension Code

```
import FilterDemoFramework

/*
  The app extension has to contain *some* code
  linking against the framework.
*/
let dummy = FilterDemoViewController.self
```

Framework: FilterDemoViewController

View controller is extension's "principal class"

Creates AUAudioUnit subclass

Creates and manages custom view

FilterDemoViewController

```
public class FilterDemoViewController : AUViewController,
  AUAudioUnitFactory, ControllerDelegate {
  public var audioUnit: AUv3FilterDemo? { … }

  public func createAudioUnitWithComponentDescription(desc:
AudioComponentDescription) throws -> AUAudioUnit {
    audioUnit = try AUv3FilterDemo(componentDescription: desc,
options: [])
    return audioUnit!
  }
}
```

FilterDemoViewController

```swift
public class FilterDemoViewController : AUViewController,
  AUAudioUnitFactory, ControllerDelegate {
  public var audioUnit: AUv3FilterDemo? { … }

  public func createAudioUnitWithComponentDescription(desc:
AudioComponentDescription) throws -> AUAudioUnit {
    audioUnit = try AUv3FilterDemo(componentDescription: desc,
options: [])
    return audioUnit!
  }
}
```

FilterDemoViewController

```
public class FilterDemoViewController : AUViewController,
  AUAudioUnitFactory, ControllerDelegate {
  public var audioUnit: AUv3FilterDemo? { … }

  public func createAudioUnitWithComponentDescription(desc:
AudioComponentDescription) throws -> AUAudioUnit {
    audioUnit = try AUv3FilterDemo(componentDescription: desc,
options: [])
    return audioUnit!
  }
}
```

FilterDemoViewController

```
public class FilterDemoViewController : AUViewController,
  AUAudioUnitFactory, ControllerDelegate {
  public var audioUnit: AUv3FilterDemo? { … }

  public func createAudioUnitWithComponentDescription(desc:
AudioComponentDescription) throws -> AUAudioUnit {
    audioUnit = try AUv3FilterDemo(componentDescription: desc,
options: [])
    return audioUnit!
  }
}
```

FilterDemoViewController

```
public class FilterDemoViewController : AUViewController,
  AUAudioUnitFactory, ControllerDelegate {
  public var audioUnit: AUv3FilterDemo? { … }

  public func createAudioUnitWithComponentDescription(desc:
AudioComponentDescription) throws -> AUAudioUnit {
    audioUnit = try AUv3FilterDemo(componentDescription: desc,
options: [])
    return audioUnit!
  }
}
```

FilterDemoViewController

```
public class FilterDemoViewController : AUViewController,
  AUAudioUnitFactory, ControllerDelegate {
  public var audioUnit: AUv3FilterDemo? { … }

  public func createAudioUnitWithComponentDescription(desc:
AudioComponentDescription) throws -> AUAudioUnit {
    audioUnit = try AUv3FilterDemo(componentDescription: desc,
options: [])
    return audioUnit!
  }
}
```

Framework: AUv3FilterDemo

AUAudioUnit subclass

```
@interface AUv3FilterDemo () {
    FilterDSPKernel kernel;
    BufferedInputBus inputBus;
    AUAudioUnitBus *outputBus;
    AUAudioUnitBusArray *inputBusArray;
    AUAudioUnitBusArray *outputBusArray;
    AUParameterTree *parameterTree;
}
```

Framework: AUv3FilterDemo

AUAudioUnit subclass

```
@interface AUv3FilterDemo () {
    FilterDSPKernel kernel;
    BufferedInputBus inputBus;
    AUAudioUnitBus *outputBus;
    AUAudioUnitBusArray *inputBusArray;
    AUAudioUnitBusArray *outputBusArray;
    AUParameterTree *parameterTree;
}
```

Framework: AUv3FilterDemo

AUAudioUnit subclass

```
@interface AUv3FilterDemo () {
    FilterDSPKernel kernel;
    BufferedInputBus inputBus;
    AUAudioUnitBus *outputBus;
    AUAudioUnitBusArray *inputBusArray;
    AUAudioUnitBusArray *outputBusArray;
    AUParameterTree *parameterTree;
}
```

Framework: AUv3FilterDemo

AUAudioUnit subclass

```
@interface AUv3FilterDemo () {
   FilterDSPKernel kernel;
   BufferedInputBus inputBus;
   AUAudioUnitBus *outputBus;
   AUAudioUnitBusArray *inputBusArray;
   AUAudioUnitBusArray *outputBusArray;
   AUParameterTree *parameterTree;
}
```

Framework: AUv3FilterDemo

AUAudioUnit subclass

```
@interface AUv3FilterDemo () {
    FilterDSPKernel kernel;
    BufferedInputBus inputBus;
    AUAudioUnitBus *outputBus;
    AUAudioUnitBusArray *inputBusArray;
    AUAudioUnitBusArray *outputBusArray;
    AUParameterTree *parameterTree;
}
```

Framework: AUv3FilterDemo

AUAudioUnit subclass

```
@interface AUv3FilterDemo () {
   FilterDSPKernel kernel;
   BufferedInputBus inputBus;
   AUAudioUnitBus *outputBus;
   AUAudioUnitBusArray *inputBusArray;
   AUAudioUnitBusArray *outputBusArray;
   AUParameterTree *parameterTree;
}
```

AUv3FilterDemo

Creating bus arrays

```
inputBus.init(defaultFormat, 8);
outputBus = [[AUAudioUnitBus alloc] initWithFormat:defaultFormat
error:nil];

inputBusArray  = [[AUAudioUnitBusArray alloc]
initWithAudioUnit:self busType:AUAudioUnitBusTypeInput  busses:
@[inputBus.bus]];
outputBusArray = [[AUAudioUnitBusArray alloc]
initWithAudioUnit:self busType:AUAudioUnitBusTypeOutput busses:
@[outputBus]];
```

AUv3FilterDemo

Creating bus arrays

```
inputBus.init(defaultFormat, 8);
outputBus = [[AUAudioUnitBus alloc] initWithFormat:defaultFormat
error:nil];
```

```
inputBusArray  = [[AUAudioUnitBusArray alloc]
initWithAudioUnit:self busType:AUAudioUnitBusTypeInput  busses:
@[inputBus.bus]];
outputBusArray = [[AUAudioUnitBusArray alloc]
initWithAudioUnit:self busType:AUAudioUnitBusTypeOutput busses:
@[outputBus]];
```

AUv3FilterDemo

Creating bus arrays

```
inputBus.init(defaultFormat, 8);
outputBus = [[AUAudioUnitBus alloc] initWithFormat:defaultFormat
error:nil];

inputBusArray  = [[AUAudioUnitBusArray alloc]
initWithAudioUnit:self busType:AUAudioUnitBusTypeInput  busses:
@[inputBus.bus]];
outputBusArray = [[AUAudioUnitBusArray alloc]
initWithAudioUnit:self busType:AUAudioUnitBusTypeOutput busses:
@[outputBus]];
```

AUv3FilterDemo

Creating bus arrays

```
inputBus.init(defaultFormat, 8);
outputBus = [[AUAudioUnitBus alloc] initWithFormat:defaultFormat
error:nil];

inputBusArray  = [[AUAudioUnitBusArray alloc]
initWithAudioUnit:self busType:AUAudioUnitBusTypeInput  busses:
@[inputBus.bus]];
outputBusArray = [[AUAudioUnitBusArray alloc]
initWithAudioUnit:self busType:AUAudioUnitBusTypeOutput busses:
@[outputBus]];
```

AUv3FilterDemo

Creating bus arrays

```
inputBus.init(defaultFormat, 8);
outputBus = [[AUAudioUnitBus alloc] initWithFormat:defaultFormat
error:nil];

inputBusArray  = [[AUAudioUnitBusArray alloc]
initWithAudioUnit:self busType:AUAudioUnitBusTypeInput  busses:
@[inputBus.bus]];
outputBusArray = [[AUAudioUnitBusArray alloc]
initWithAudioUnit:self busType:AUAudioUnitBusTypeOutput busses:
@[outputBus]];
```

AUv3FilterDemo

Creating bus arrays

```
inputBus.init(defaultFormat, 8);
outputBus = [[AUAudioUnitBus alloc] initWithFormat:defaultFormat
error:nil];


inputBusArray  = [[AUAudioUnitBusArray alloc]
initWithAudioUnit:self busType:AUAudioUnitBusTypeInput  busses:
@[inputBus.bus]];
outputBusArray = [[AUAudioUnitBusArray alloc]
initWithAudioUnit:self busType:AUAudioUnitBusTypeOutput busses:
@[outputBus]];
```

AUv3FilterDemo

Creating parameters

```
AUParameter *cutoffParam =
  [AUParameterTree createParameterWithIdentifier:@"cutoff"
    name:@“Cutoff" address:FilterParamCutoff
    min:12.0 max:20000.0
    unit:kAudioUnitParameterUnit_Hertz unitName:nil
    flags:0 valueStrings:nil dependentParameters:nil];
AUParameter *resonanceParam =
  [AUParameterTree createParameterWithIdentifier:@"resonance"
    name:@“Resonance" address:FilterParamResonance
    …
```

AUv3FilterDemo

Creating parameters

```
AUParameter *cutoffParam =
  [AUParameterTree createParameterWithIdentifier:@"cutoff"
    name:@“Cutoff" address:FilterParamCutoff
    min:12.0 max:20000.0
    unit:kAudioUnitParameterUnit_Hertz unitName:nil
    flags:0 valueStrings:nil dependentParameters:nil];
AUParameter *resonanceParam =
  [AUParameterTree createParameterWithIdentifier:@"resonance"
    name:@“Resonance" address:FilterParamResonance
    …
```

AUv3FilterDemo

Creating parameters

```
AUParameter *cutoffParam =
  [AUParameterTree createParameterWithIdentifier:@"cutoff"
    name:@“Cutoff" address:FilterParamCutoff
    min:12.0 max:20000.0
    unit:kAudioUnitParameterUnit_Hertz unitName:nil
    flags:0 valueStrings:nil dependentParameters:nil];
AUParameter *resonanceParam =
  [AUParameterTree createParameterWithIdentifier:@"resonance"
    name:@“Resonance" address:FilterParamResonance
    …
```

AUv3FilterDemo

Creating parameters

```
AUParameter *cutoffParam =
  [AUParameterTree createParameterWithIdentifier:@"cutoff"
    name:@“Cutoff" address:FilterParamCutoff
    min:12.0 max:20000.0
    unit:kAudioUnitParameterUnit_Hertz unitName:nil
    flags:0 valueStrings:nil dependentParameters:nil];
AUParameter *resonanceParam =
  [AUParameterTree createParameterWithIdentifier:@"resonance"
    name:@“Resonance" address:FilterParamResonance
    …
```

AUv3FilterDemo

Creating the parameter tree

```
parameterTree = [AUParameterTree createTreeWithChildren:
  @[cutoffParam, resonanceParam]];

parameterTree.implementorValueObserver =
  ^(AUParameter *param, AUValue value) {
    filterDSPKernel->setParameter(param.address, value);
  };

parameterTree.implementorValueProvider =
  ^(AUParameter *param) {
    return filterDSPKernel->getParameter(param.address);
  };
```

AUv3FilterDemo

Creating the parameter tree

```
parameterTree = [AUParameterTree createTreeWithChildren:
  @[cutoffParam, resonanceParam]];

parameterTree.implementorValueObserver =
  ^(AUParameter *param, AUValue value) {
    filterDSPKernel->setParameter(param.address, value);
  };

parameterTree.implementorValueProvider =
  ^(AUParameter *param) {
    return filterDSPKernel->getParameter(param.address);
  };
```

AUv3FilterDemo

Creating the parameter tree

```
parameterTree = [AUParameterTree createTreeWithChildren:
  @[cutoffParam, resonanceParam]];

parameterTree.implementorValueObserver =
  ^(AUParameter *param, AUValue value) {
    filterDSPKernel->setParameter(param.address, value);
  };

parameterTree.implementorValueProvider =
  ^(AUParameter *param) {
    return filterDSPKernel->getParameter(param.address);
  };
```

AUv3FilterDemo

Creating the parameter tree

```
parameterTree = [AUParameterTree createTreeWithChildren:
  @[cutoffParam, resonanceParam]];
parameterTree.implementorValueObserver =
  ^(AUParameter *param, AUValue value) {
    filterDSPKernel->setParameter(param.address, value);
  };
parameterTree.implementorValueProvider =
  ^(AUParameter *param) {
    return filterDSPKernel->getParameter(param.address);
  };
```

AUv3FilterDemo

Creating the parameter tree

```
parameterTree = [AUParameterTree createTreeWithChildren:
  @[cutoffParam, resonanceParam]];

parameterTree.implementorValueObserver =
  ^(AUParameter *param, AUValue value) {
    filterDSPKernel->setParameter(param.address, value);
  };

parameterTree.implementorValueProvider =
  ^(AUParameter *param) {
    return filterDSPKernel->getParameter(param.address);
  };
```

AUv3FilterDemo

Creating the parameter tree

```
parameterTree = [AUParameterTree createTreeWithChildren:
  @[cutoffParam, resonanceParam]];

parameterTree.implementorValueObserver =
  ^(AUParameter *param, AUValue value) {
    filterDSPKernel->setParameter(param.address, value);
  };

parameterTree.implementorValueProvider =
  ^(AUParameter *param) {
    return filterDSPKernel->getParameter(param.address);
  };
```

AUv3FilterDemo

Preparing to render

```
- (BOOL)allocateRenderResourcesAndReturnError:(NSError **)outError
{
  if (![super allocateRenderResourcesAndReturnError: outError]) {
    return NO; }
  inputBus.allocateRenderResources(self.maximumFramesToRender);
  kernel.init(outputBus.format.channelCount,
    outputBus.format.sampleRate);
  kernel.reset();
```

AUv3FilterDemo

Preparing to render

```
- (BOOL)allocateRenderResourcesAndReturnError:(NSError **)outError
{
  if (![super allocateRenderResourcesAndReturnError: outError]) {
    return NO; }

  inputBus.allocateRenderResources(self.maximumFramesToRender);

  kernel.init(outputBus.format.channelCount,
    outputBus.format.sampleRate);

  kernel.reset();
```

AUv3FilterDemo

Preparing to render

```
- (BOOL)allocateRenderResourcesAndReturnError:(NSError **)outError
{
  if (![super allocateRenderResourcesAndReturnError: outError]) {
    return NO; }
  inputBus.allocateRenderResources(self.maximumFramesToRender);
  kernel.init(outputBus.format.channelCount,
    outputBus.format.sampleRate);
  kernel.reset();
```

AUv3FilterDemo

Preparing to render

```
- (BOOL)allocateRenderResourcesAndReturnError:(NSError **)outError
{
  if (![super allocateRenderResourcesAndReturnError: outError]) {
    return NO; }
  inputBus.allocateRenderResources(self.maximumFramesToRender);
  kernel.init(outputBus.format.channelCount,
    outputBus.format.sampleRate);
  kernel.reset();
```

AUv3FilterDemo

Preparing to render

```
- (BOOL)allocateRenderResourcesAndReturnError:(NSError **)outError
{
  if (![super allocateRenderResourcesAndReturnError: outError]) {
    return NO; }
  inputBus.allocateRenderResources(self.maximumFramesToRender);
  kernel.init(outputBus.format.channelCount,
    outputBus.format.sampleRate);
  kernel.reset();
```

AUv3FilterDemo

Preparing to render

```
- (BOOL)allocateRenderResourcesAndReturnError:(NSError **)outError
{
  if (![super allocateRenderResourcesAndReturnError: outError]) {
    return NO; }
  inputBus.allocateRenderResources(self.maximumFramesToRender);
  kernel.init(outputBus.format.channelCount,
    outputBus.format.sampleRate);
  kernel.reset();
```

AUv3FilterDemo

Preparing to render

```
- (BOOL)allocateRenderResourcesAndReturnError:(NSError **)outError
{
  if (![super allocateRenderResourcesAndReturnError: outError]) {
    return NO; }
  inputBus.allocateRenderResources(self.maximumFramesToRender);
  kernel.init(outputBus.format.channelCount,
    outputBus.format.sampleRate);
  kernel.reset();
```

AUv3FilterDemo

Preparing to render

```
- (BOOL)allocateRenderResourcesAndReturnError:(NSError **)outError
{
  if (![super allocateRenderResourcesAndReturnError: outError]) {
    return NO; }
  inputBus.allocateRenderResources(self.maximumFramesToRender);
  kernel.init(outputBus.format.channelCount,
    outputBus.format.sampleRate);
  kernel.reset();
```

AUv3FilterDemo

Cleaning up

```
- (void)deallocateRenderResources
{
  [super deallocateRenderResources];
  inputBus.deallocateRenderResources();
  …
```

AUv3FilterDemo

Rendering

```
- (AUInternalRenderBlock)internalRenderBlock
{
  // Capture properties into locals.
  __block FilterDSPKernel *state = &kernel;
  __block BufferedInputBus *input = &inputBus;
```

AUv3FilterDemo

Rendering

```
return ^AUAudioUnitStatus(
  AudioUnitRenderActionFlags *actionFlags,
  const AudioTimeStamp       *timestamp,
  AVAudioFrameCount           frameCount,
  NSInteger                   outputBusNumber,
  AudioBufferList            *outputData,
  const AURenderEvent        *realtimeEventListHead,
  AURenderPullInputBlock      pullInputBlock)
{
  …
```

AUv3FilterDemo

Rendering

```
return ^AUAudioUnitStatus(
   AudioUnitRenderActionFlags *actionFlags,
   const AudioTimeStamp       *timestamp,
   AVAudioFrameCount           frameCount,
   NSInteger                   outputBusNumber,
   AudioBufferList            *outputData,
   const AURenderEvent        *realtimeEventListHead,
   AURenderPullInputBlock      pullInputBlock)
{
   …
```

AUv3FilterDemo

Rendering

```
return ^AUAudioUnitStatus(
  AudioUnitRenderActionFlags *actionFlags,
  const AudioTimeStamp       *timestamp,
  AVAudioFrameCount           frameCount,
  NSInteger                   outputBusNumber,
  AudioBufferList            *outputData,
  const AURenderEvent        *realtimeEventListHead,
  AURenderPullInputBlock      pullInputBlock)
{
  …
```

AUv3FilterDemo

Rendering

```
return ^AUAudioUnitStatus(
  AudioUnitRenderActionFlags *actionFlags,
  const AudioTimeStamp       *timestamp,
  AVAudioFrameCount           frameCount,
  NSInteger                   outputBusNumber,
  AudioBufferList            *outputData,
  const AURenderEvent        *realtimeEventListHead,
  AURenderPullInputBlock      pullInputBlock)
{
  …
```

AUv3FilterDemo

Rendering

```
return ^AUAudioUnitStatus(
  AudioUnitRenderActionFlags *actionFlags,
  const AudioTimeStamp       *timestamp,
  AVAudioFrameCount           frameCount,
  NSInteger                   outputBusNumber,
  AudioBufferList            *outputData,
  const AURenderEvent        *realtimeEventListHead,
  AURenderPullInputBlock      pullInputBlock)
{
  …
```

AUv3FilterDemo

Rendering

```
return ^AUAudioUnitStatus(
  AudioUnitRenderActionFlags *actionFlags,
  const AudioTimeStamp       *timestamp,
  AVAudioFrameCount           frameCount,
  NSInteger                   outputBusNumber,
  AudioBufferList            *outputData,
  const AURenderEvent        *realtimeEventListHead,
  AURenderPullInputBlock      pullInputBlock)
{
  …
```

AUv3FilterDemo

Rendering

```
return ^AUAudioUnitStatus(
  AudioUnitRenderActionFlags *actionFlags,
  const AudioTimeStamp       *timestamp,
  AVAudioFrameCount           frameCount,
  NSInteger                   outputBusNumber,
  AudioBufferList            *outputData,
  const AURenderEvent        *realtimeEventListHead,
  AURenderPullInputBlock      pullInputBlock)
{
  …
```

AUv3FilterDemo

Rendering

```
return ^AUAudioUnitStatus(
  AudioUnitRenderActionFlags *actionFlags,
  const AudioTimeStamp       *timestamp,
  AVAudioFrameCount           frameCount,
  NSInteger                   outputBusNumber,
  AudioBufferList            *outputData,
  const AURenderEvent        *realtimeEventListHead,
  AURenderPullInputBlock      pullInputBlock)
{
  …
```

AUv3FilterDemo

Rendering

```
return ^AUAudioUnitStatus(
  AudioUnitRenderActionFlags *actionFlags,
  const AudioTimeStamp       *timestamp,
  AVAudioFrameCount           frameCount,
  NSInteger                   outputBusNumber,
  AudioBufferList            *outputData,
  const AURenderEvent        *realtimeEventListHead,
  AURenderPullInputBlock      pullInputBlock)
{
  …
```

AUv3FilterDemo

Rendering

```
return ^AUAudioUnitStatus(
  AudioUnitRenderActionFlags *actionFlags,
  const AudioTimeStamp       *timestamp,
  AVAudioFrameCount           frameCount,
  NSInteger                   outputBusNumber,
  AudioBufferList            *outputData,
  const AURenderEvent        *realtimeEventListHead,
  AURenderPullInputBlock      pullInputBlock)
{
  …
```

AUv3FilterDemo

Rendering

```
input->pullInput(&pullFlags, timestamp, frameCount, 0,
  pullInputBlock);

AudioBufferList *inAudioBufferList =
  input->mutableAudioBufferList;

state->setBuffers(inAudioBufferList, outAudioBufferList);

state->processWithEvents(timestamp, frameCount,
  realtimeEventListHead);
```

AUv3FilterDemo

Rendering

```
input->pullInput(&pullFlags, timestamp, frameCount, 0,
  pullInputBlock);

AudioBufferList *inAudioBufferList =
  input->mutableAudioBufferList;

state->setBuffers(inAudioBufferList, outAudioBufferList);

state->processWithEvents(timestamp, frameCount,
  realtimeEventListHead);
```

AUv3FilterDemo

Rendering

```
input->pullInput(&pullFlags, timestamp, frameCount, 0,
  pullInputBlock);

AudioBufferList *inAudioBufferList =
  input->mutableAudioBufferList;

state->setBuffers(inAudioBufferList, outAudioBufferList);

state->processWithEvents(timestamp, frameCount,
  realtimeEventListHead);
```

AUv3FilterDemo

Rendering

```
input->pullInput(&pullFlags, timestamp, frameCount, 0,
  pullInputBlock);
AudioBufferList *inAudioBufferList =
  input->mutableAudioBufferList;
state->setBuffers(inAudioBufferList, outAudioBufferList);
state->processWithEvents(timestamp, frameCount,
  realtimeEventListHead);
```

AUv3FilterDemo

Rendering

```
input->pullInput(&pullFlags, timestamp, frameCount, 0,
  pullInputBlock);
AudioBufferList *inAudioBufferList =
  input->mutableAudioBufferList;
state->setBuffers(inAudioBufferList, outAudioBufferList);
state->processWithEvents(timestamp, frameCount,
  realtimeEventListHead);
```

AUv3FilterDemo

Rendering

```
input->pullInput(&pullFlags, timestamp, frameCount, 0,
  pullInputBlock);

AudioBufferList *inAudioBufferList =
  input->mutableAudioBufferList;

state->setBuffers(inAudioBufferList, outAudioBufferList);

state->processWithEvents(timestamp, frameCount,
  realtimeEventListHead);
```

Demo

AUv3FilterDemo

Michael Hopkins
Purveyor of Pixels

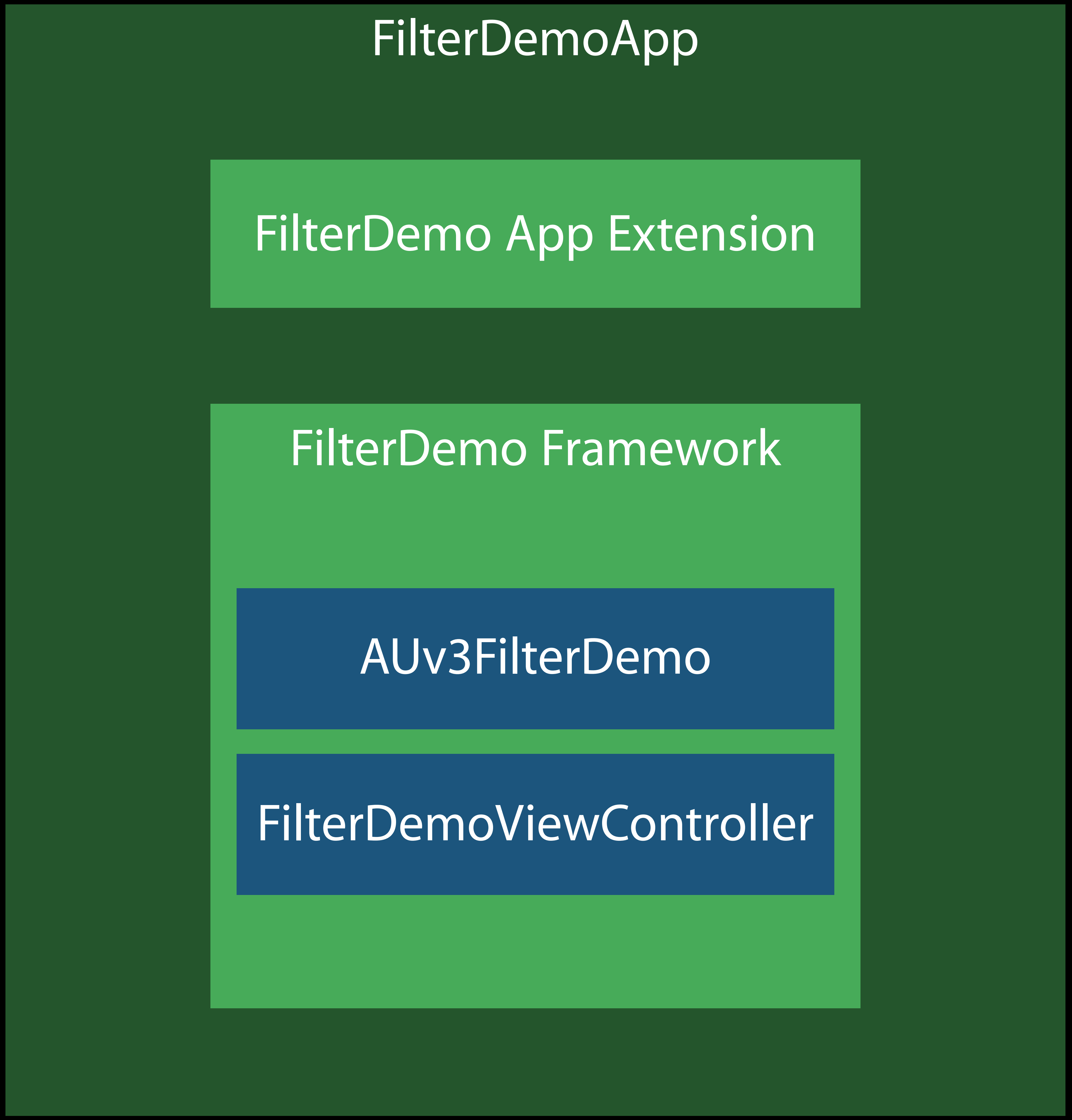
Containing App

Plug-in "vehicle"

Rapid iteration in development

Extra functionality, e.g.

- Playback engine
- Touch controller
- Documentation/help



Creating an Extension

OS X in-process framework

- Caution: Swift ABI subject to change

Xcode template planned

- For now, copy from FilterDemo

Modernized API

AUAudioUnit

Properties: v2

Scope/element based properties, `void *` access methods

- `AudioUnitGetProperty(audioUnit, propertyID, scope, element,`
  `data, dataSize)`
- `AudioUnitSetProperty`
- Awkward from Swift

Properties: v3

Dot syntax in ObjC and Swift, KVC/KVO:

```
au.maximumFramesToRender = 4096

let maxFrames = au.valueForKey(“maximumFramesToRender”) as? Int

mixerAU.addObserver(self, forKeyPath: "inputBusses", options: nil,
    context: nil)

mixerAU.inputBusses.addObserverToAllBusses(self,
    forKeyPath: "format", options: nil, context: nil)
```

Busses

Objects

- AUAudioUnitBusArray
- AUAudioUnitBus

```
let fmt: AVAudioFormat = au.inputBusses[0].format
let sampleRate: Double = fmt.sampleRate
au.outputBusses[0].name = “Reverb send”
```

Parameters: v2

Scope/element/ID

- Unweildy tuple
- Not enough bits for complex AU's

```
AudioUnitGetParameter(audioUnit, paramID, scope, element, value)
AudioUnitSetParameter(audioUnit, paramID, scope, element, value)
```

Complex AUEventListener API

AUAudioUnit.parameterTree

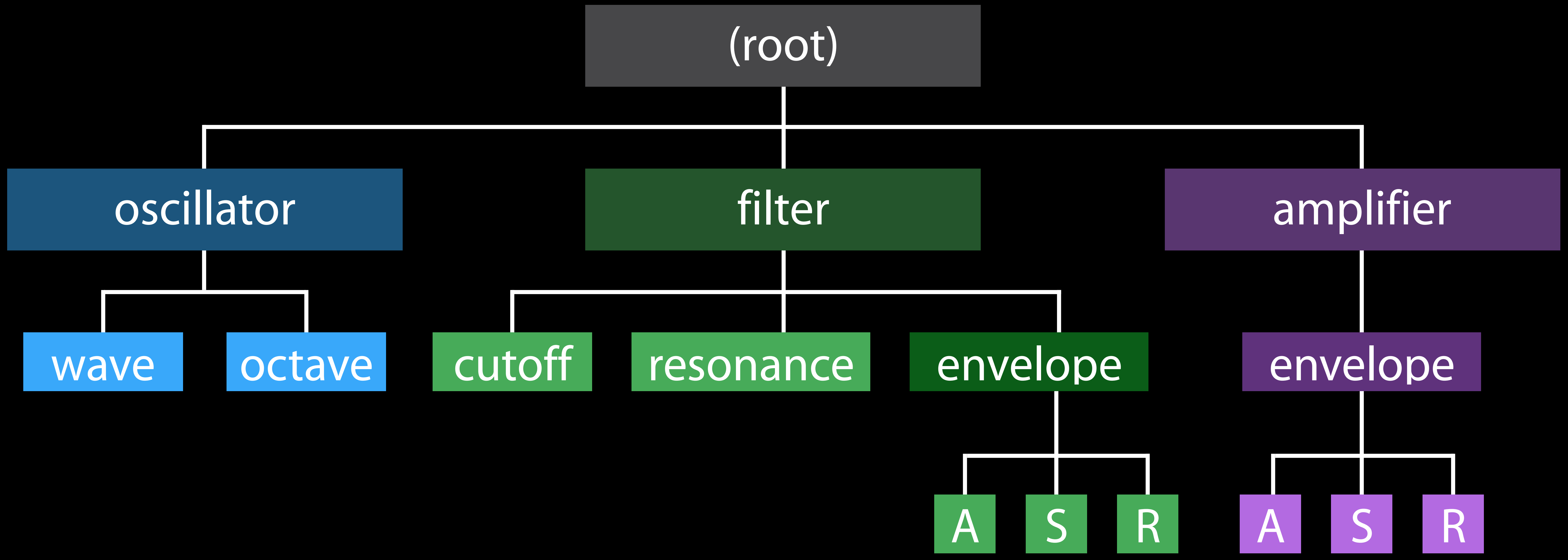
Nodes

- Groups
- Parameters
- Permanent,
  unique string ID's

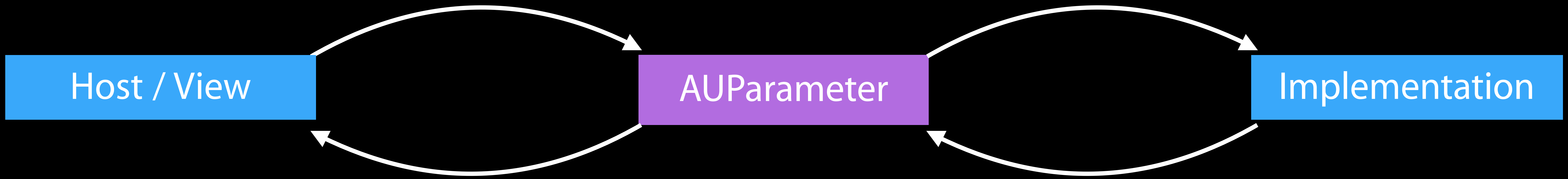
Key path (KVC)

- `oscillator.wave`
- `filter.envelope.attack`

Parameters also have numeric addresses (transient)

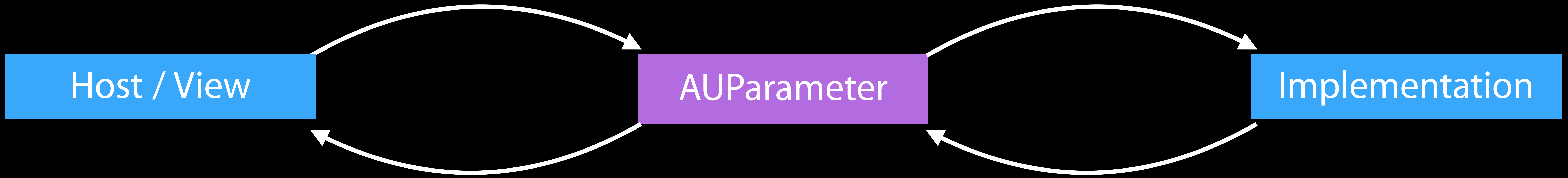


Parameter Wiring



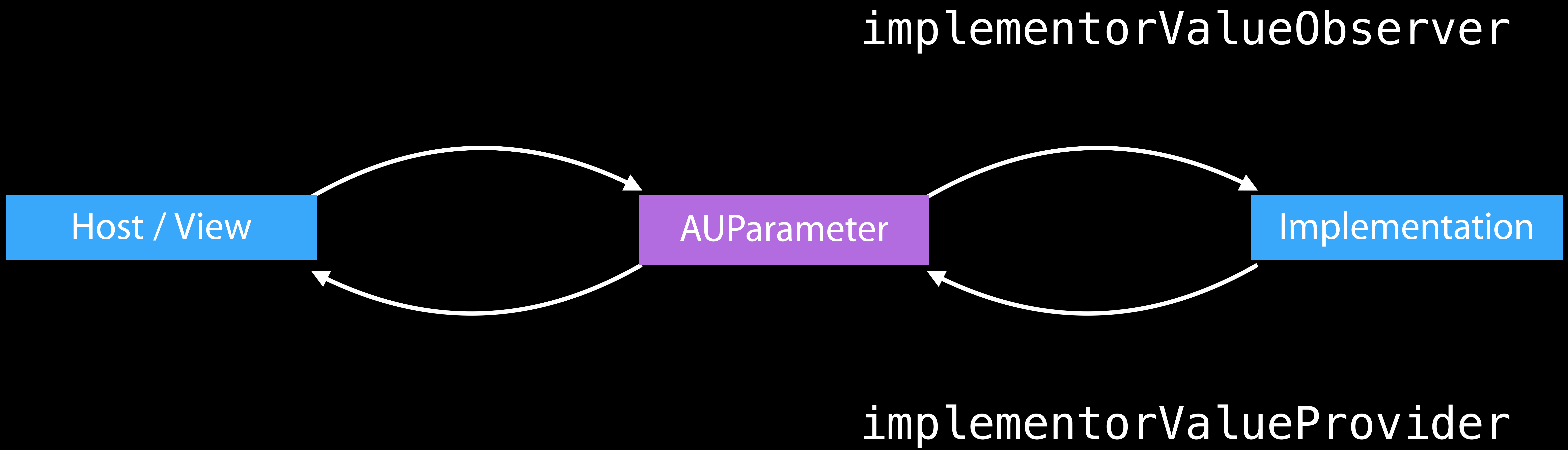
Parameter Wiring

```
   param.value = x
   param.setValue(
x, originator: token)
```

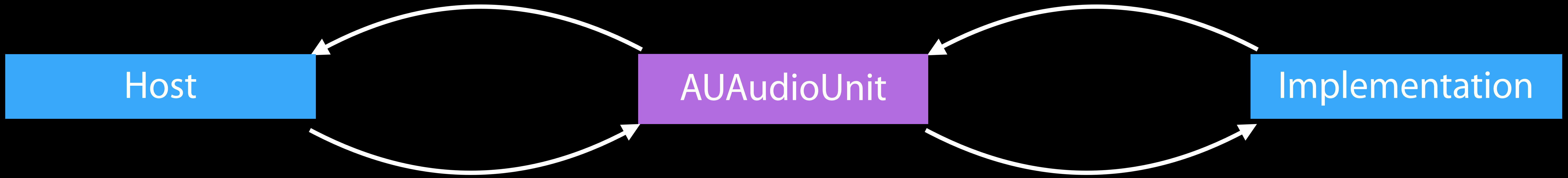


```
(^AUParameterObserver)
(AUParameterAddress
address, AUValue value)
```

Parameter Wiring

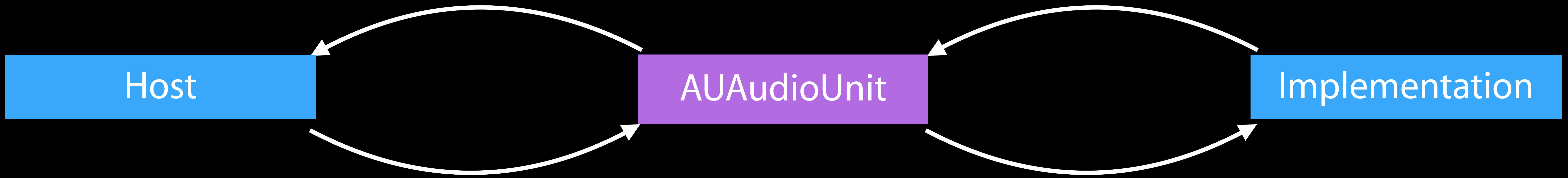


Parameter Scheduling



Parameter Scheduling

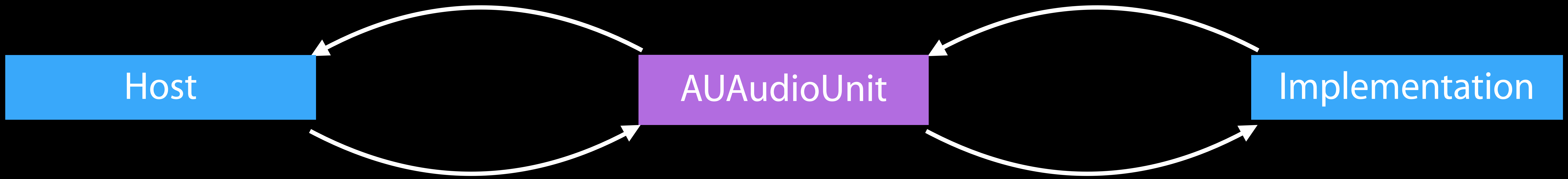
```
        doSchedule =
au.scheduleParameterBlock
```



```
doSchedule(when, rampDurationFrames,
           paramAddress,
           value)
```

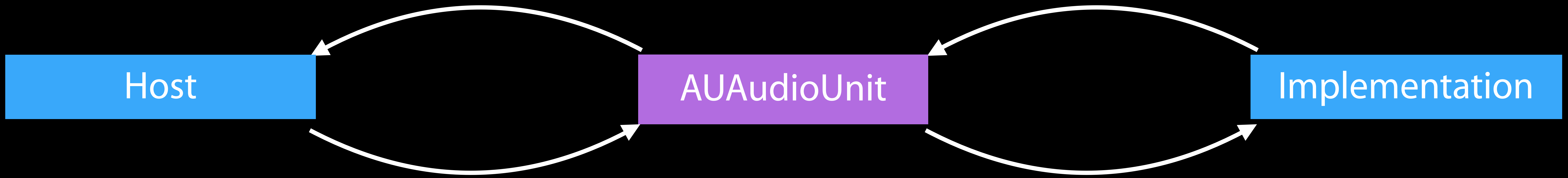
Parameter Scheduling

```
internalRender =
au.internalRenderBlock
```



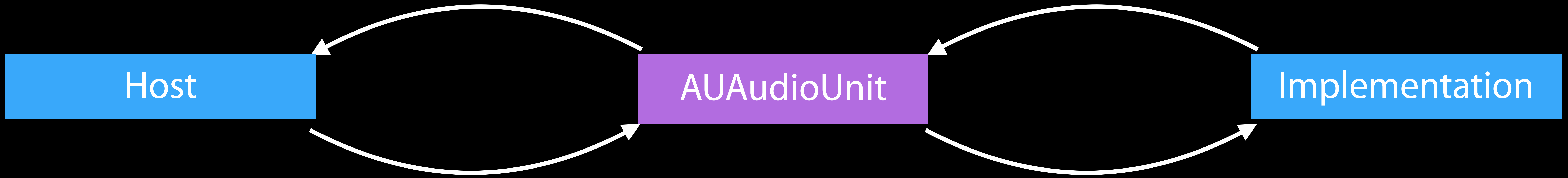
```
internalRender(… AURenderEvent
realtimeEventListHead …)
```

MIDI Events



MIDI Events

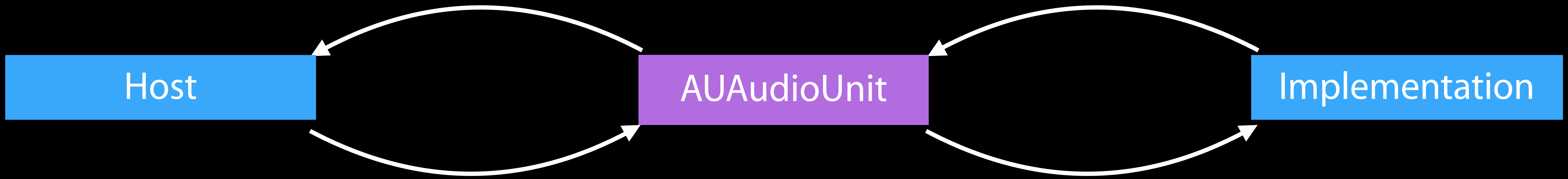
```
     scheduleMIDI =
au.scheduleMIDIEventBlock
```



```
scheduleMIDI(when, cable,
    numBytes, bytePtr)
```

MIDI Events

```
internalRender =
au.internalRenderBlock
```



```
internalRender(… AURenderEvent
realtimeEventListHead …)
```

Rendering

Pull model

v2: AU has connection or input callback

v3: Input always from callback

Otherwise, functionally identical

- Efficient bridging

Calling the Render Block

For hosts

Fetch the block when allocating render resources:

```
au.allocateRenderResources()
renderBlock = au.renderBlock
```

Call it to render:

```
renderBlock(…)
```

Render Block: Output Buffers

Host provides `AudioBufferList` for the Audio Unit's output

`bufferList.mBuffers[i].mData` can be null

- AU must replace this with an internally-owned buffer
- Buffer must remain valid until next render cycle

Render Block: Input Buffers

Host provides `AURenderPullInputBlock`

AU calls block for input

AU must supply valid `AudioBufferList` (non-null **`mData`** pointers)

Host may replace **`mData`** pointers

- Must guarantee that memory remains valid until next render cycle

Goal—Avoid copying

Rendering

Realtime thread context

- Cannot allocate memory
- Cannot make blocking calls

Using/implementing render blocks

- Never capture self: ObjC runtime unsafe
- Swift unsafe too
- To capture state: use pointer to C struct, C++ object

Apple Music Creation Apps

Audio Unit Extensions

Alec Little
Product Designer

Upcoming Support for Audio Unit Extensions

GarageBand iOS

GarageBand Mac

Logic Pro X

Mainstage

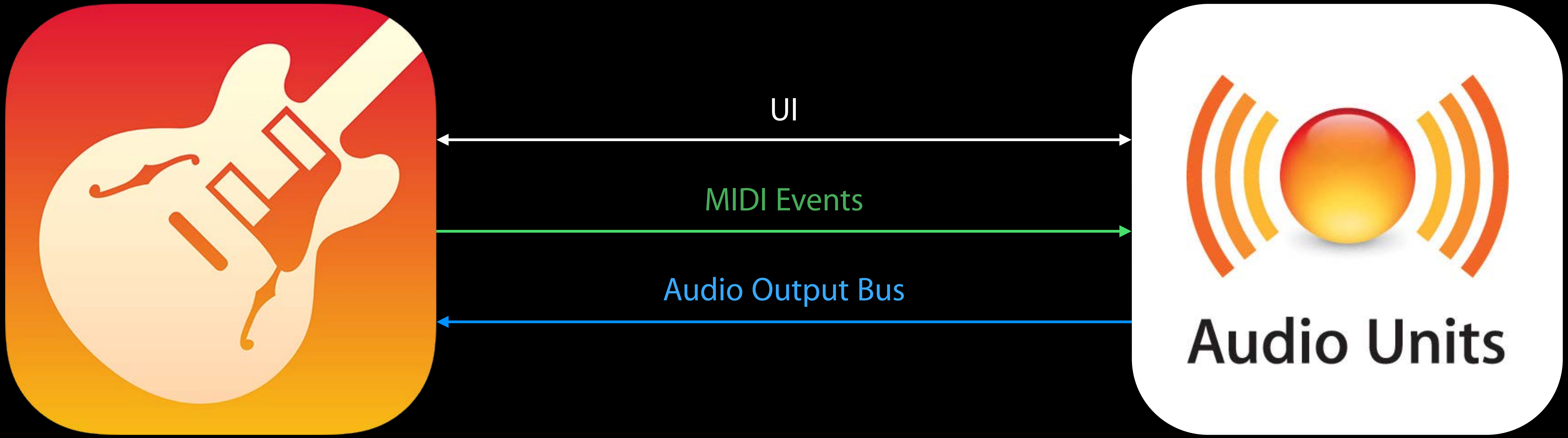
GarageBand iOS

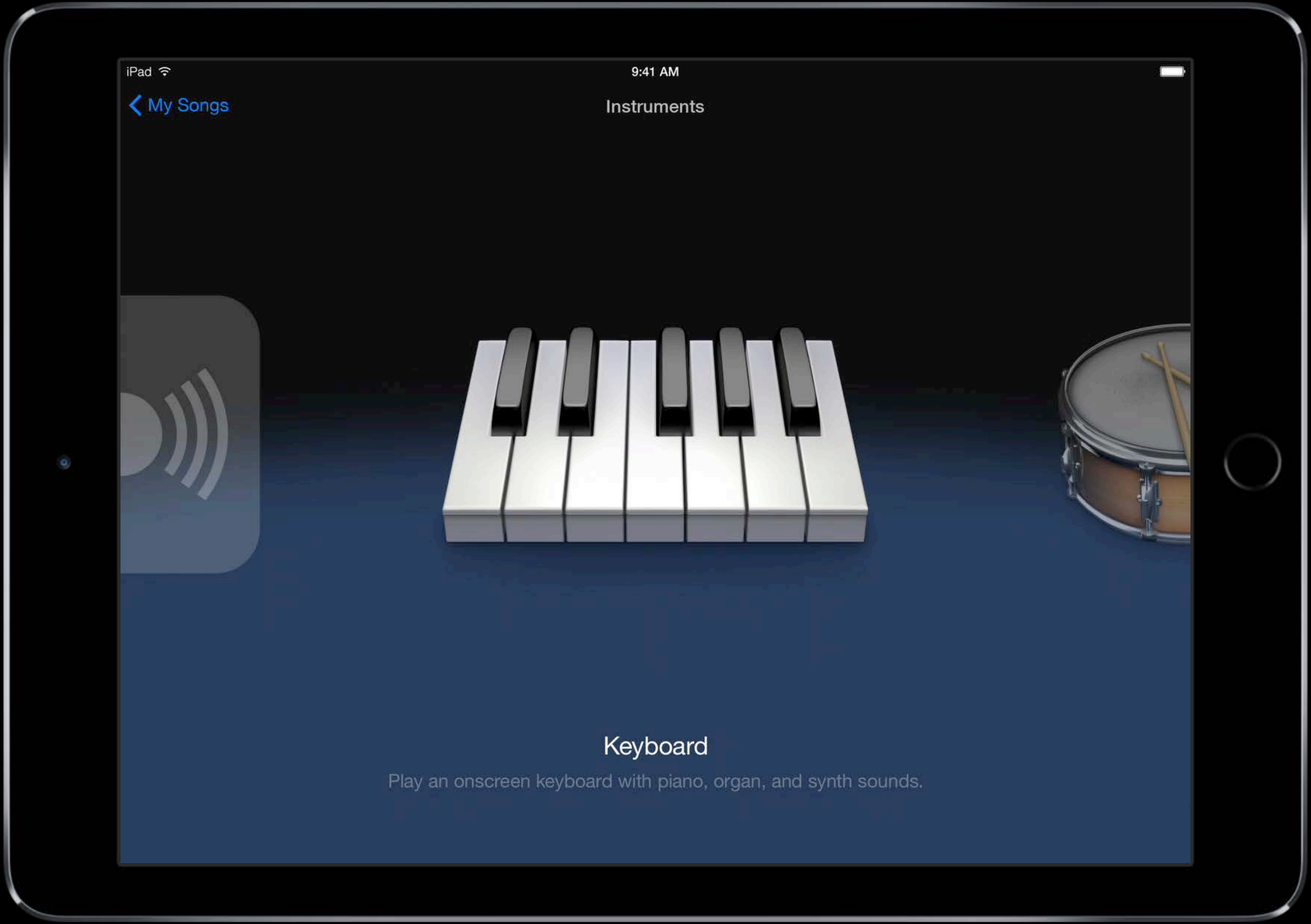
Preliminary Audio Unit Extension UI

Preliminary Audio Unit Extension UI

AU Instruments

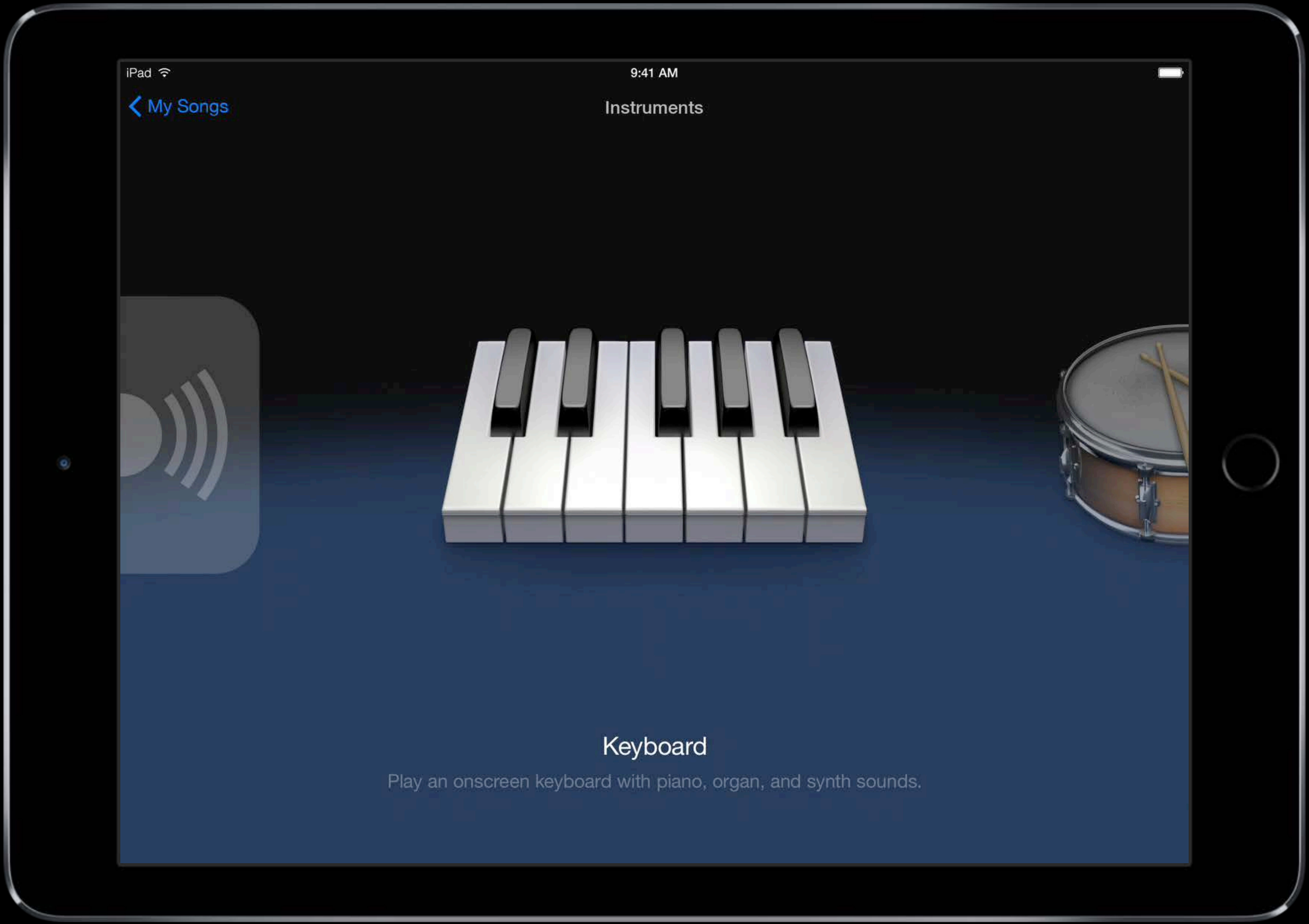
AU Instruments





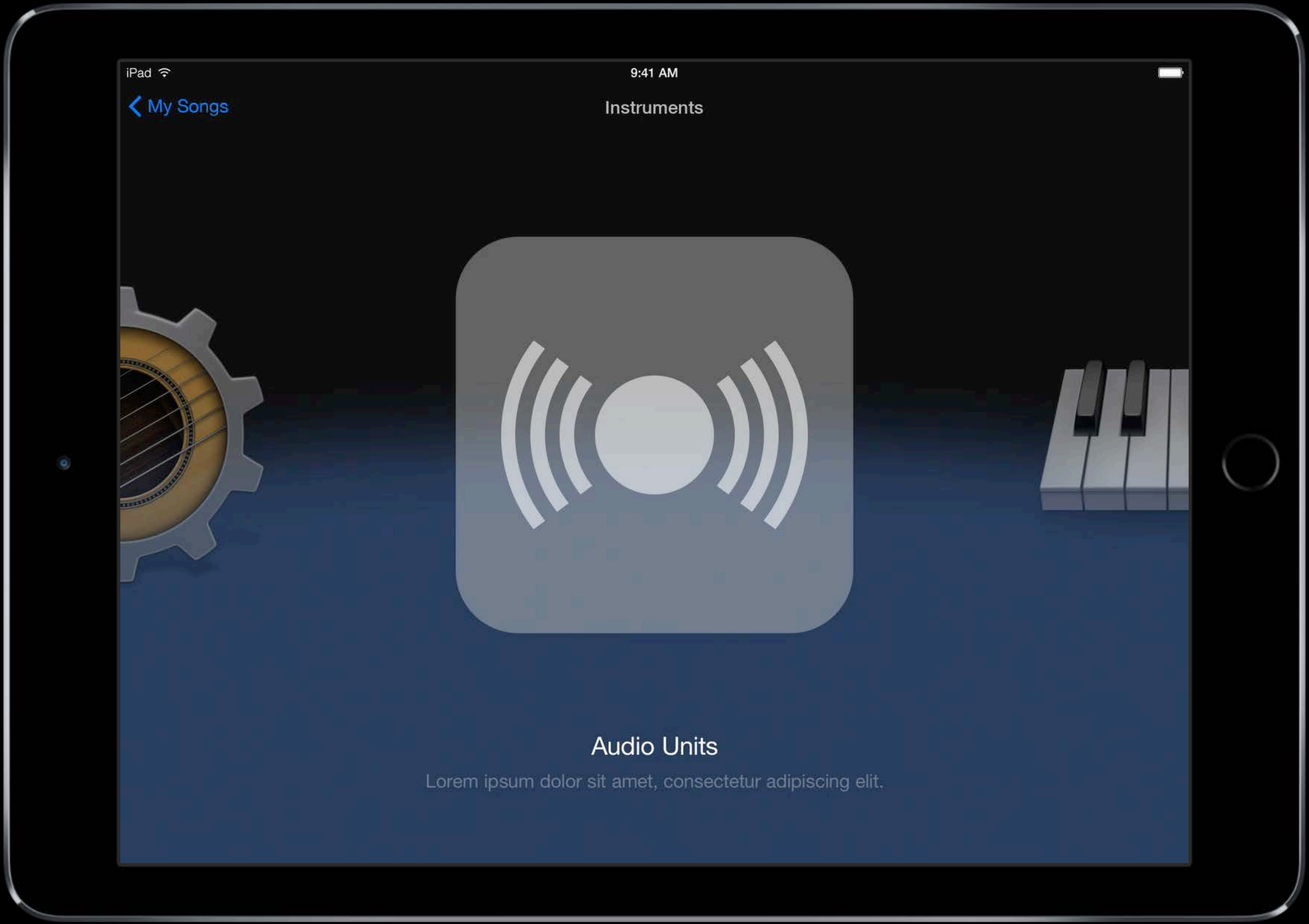
Keyboard

Play an onscreen keyboard with piano, organ, and synth sounds.



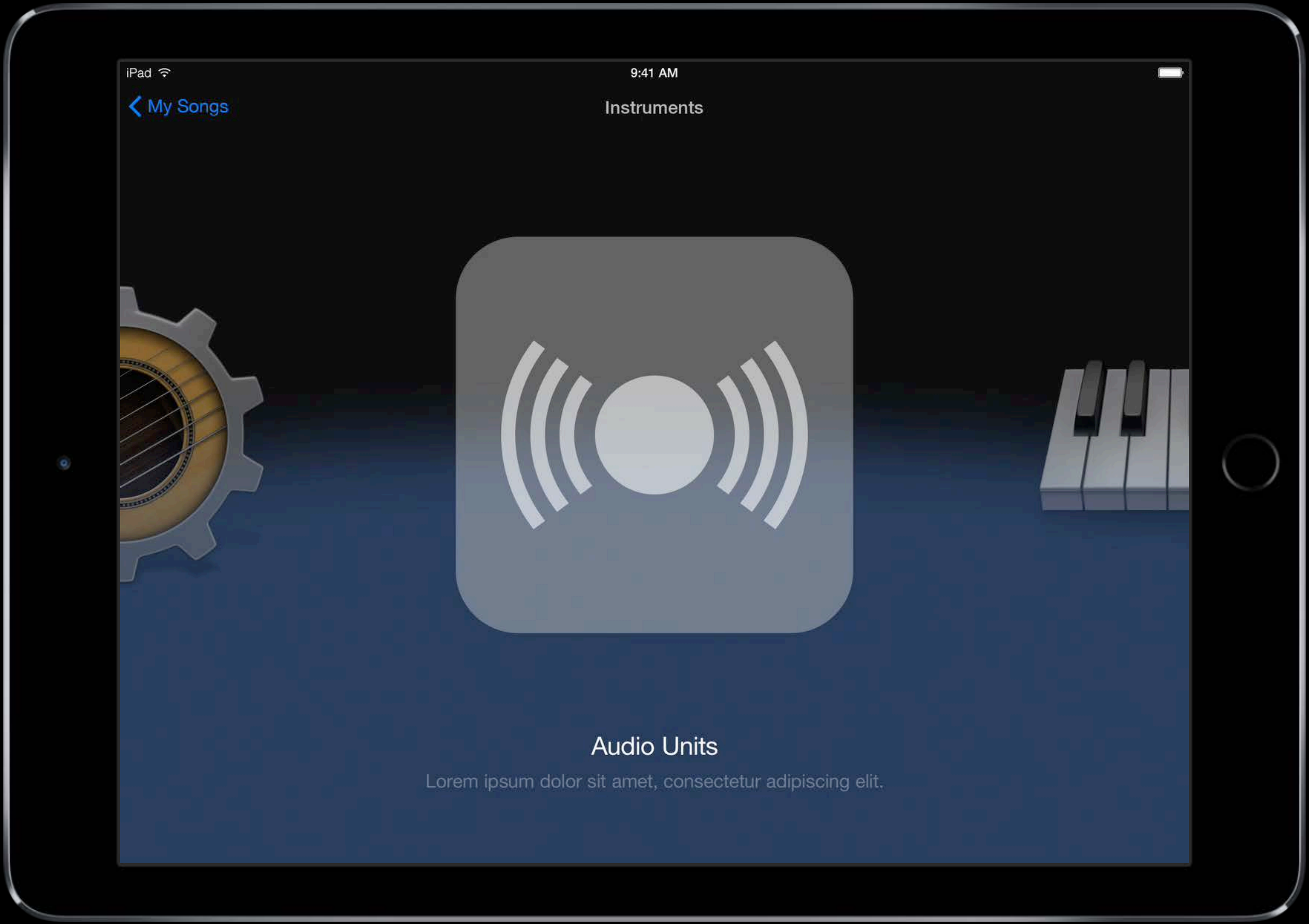
Keyboard

Play an onscreen keyboard with piano, organ, and synth sounds.



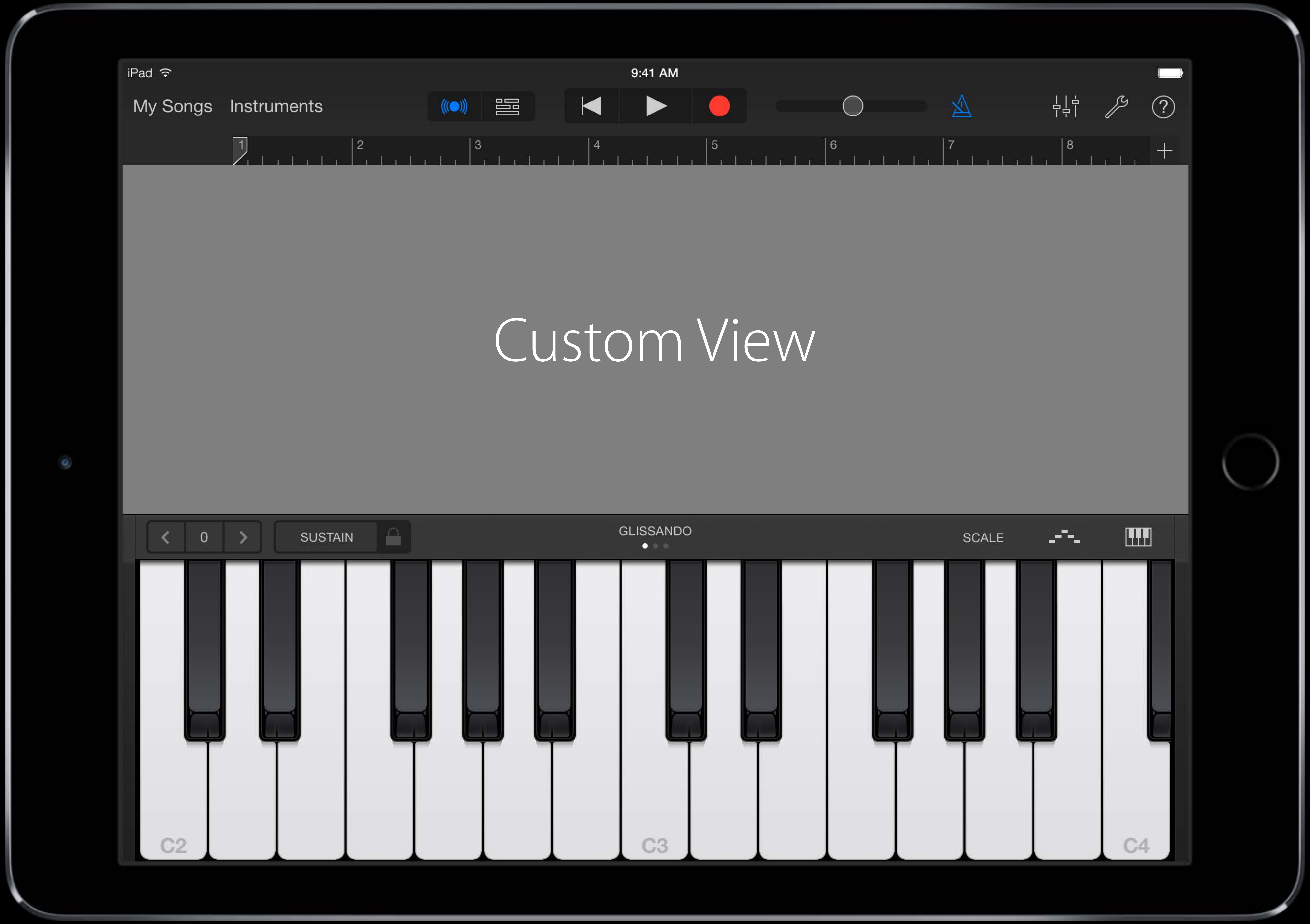
Audio Units

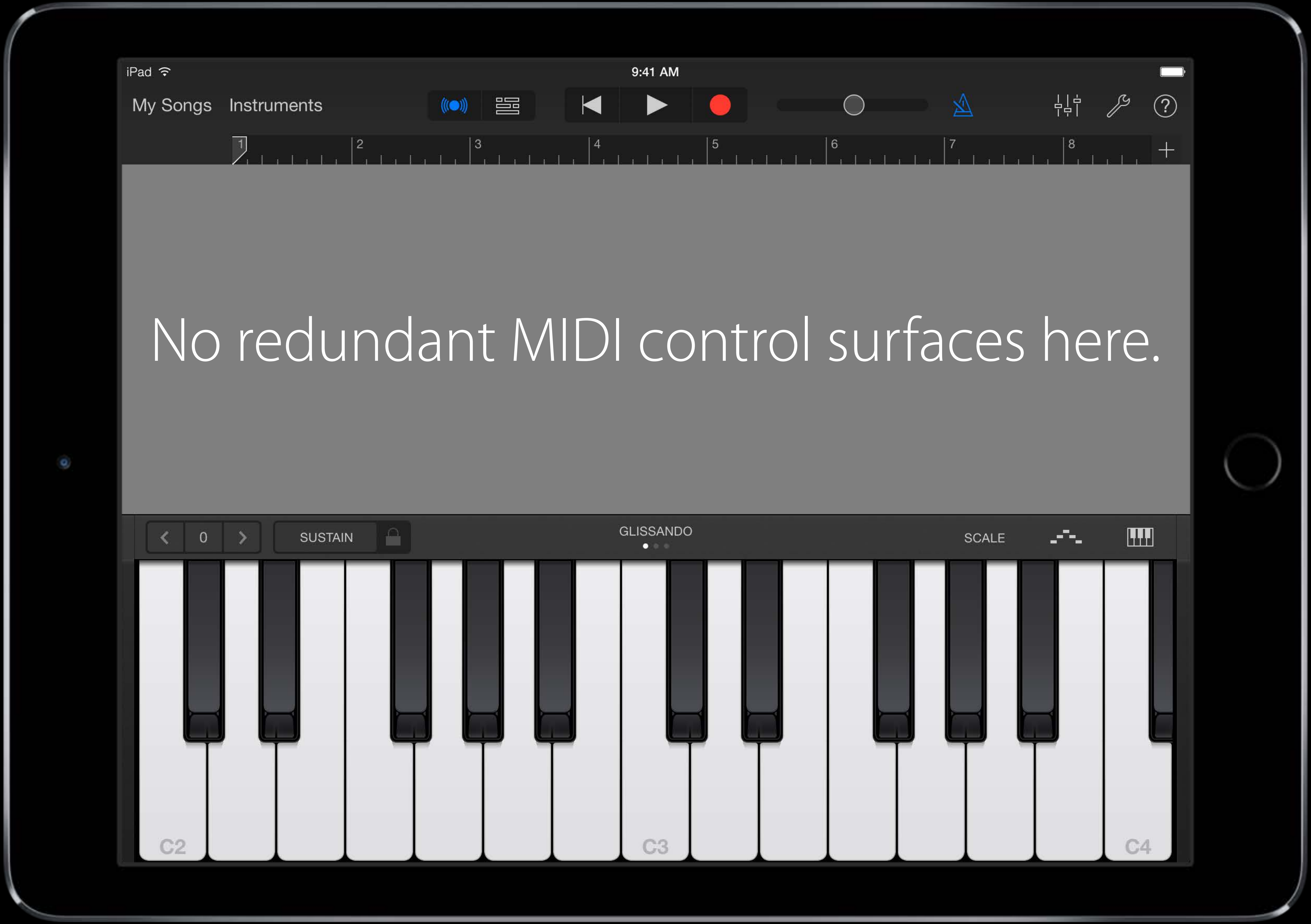
Lorem ipsum dolor sit amet, consectetur adipiscing elit.

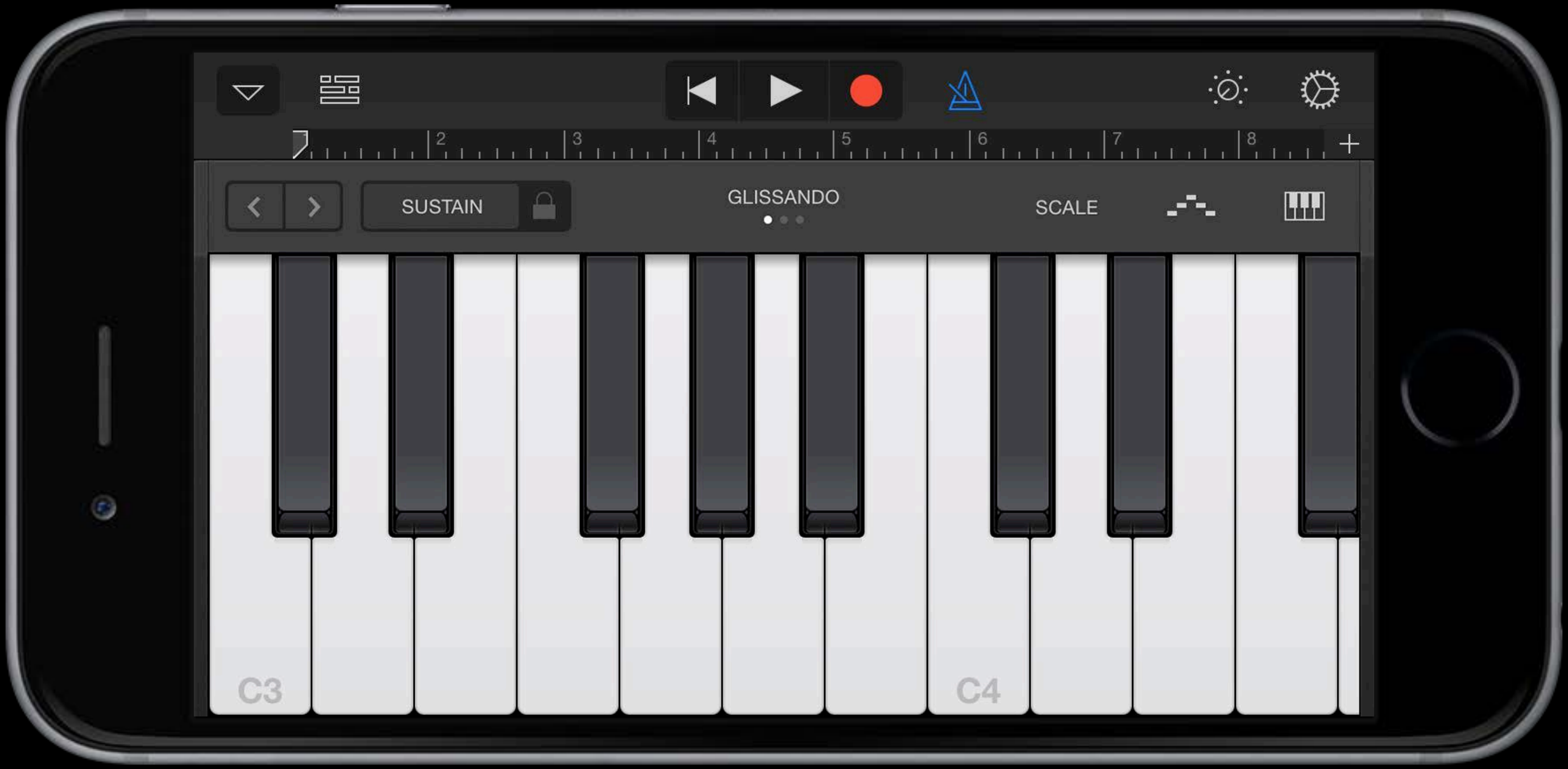


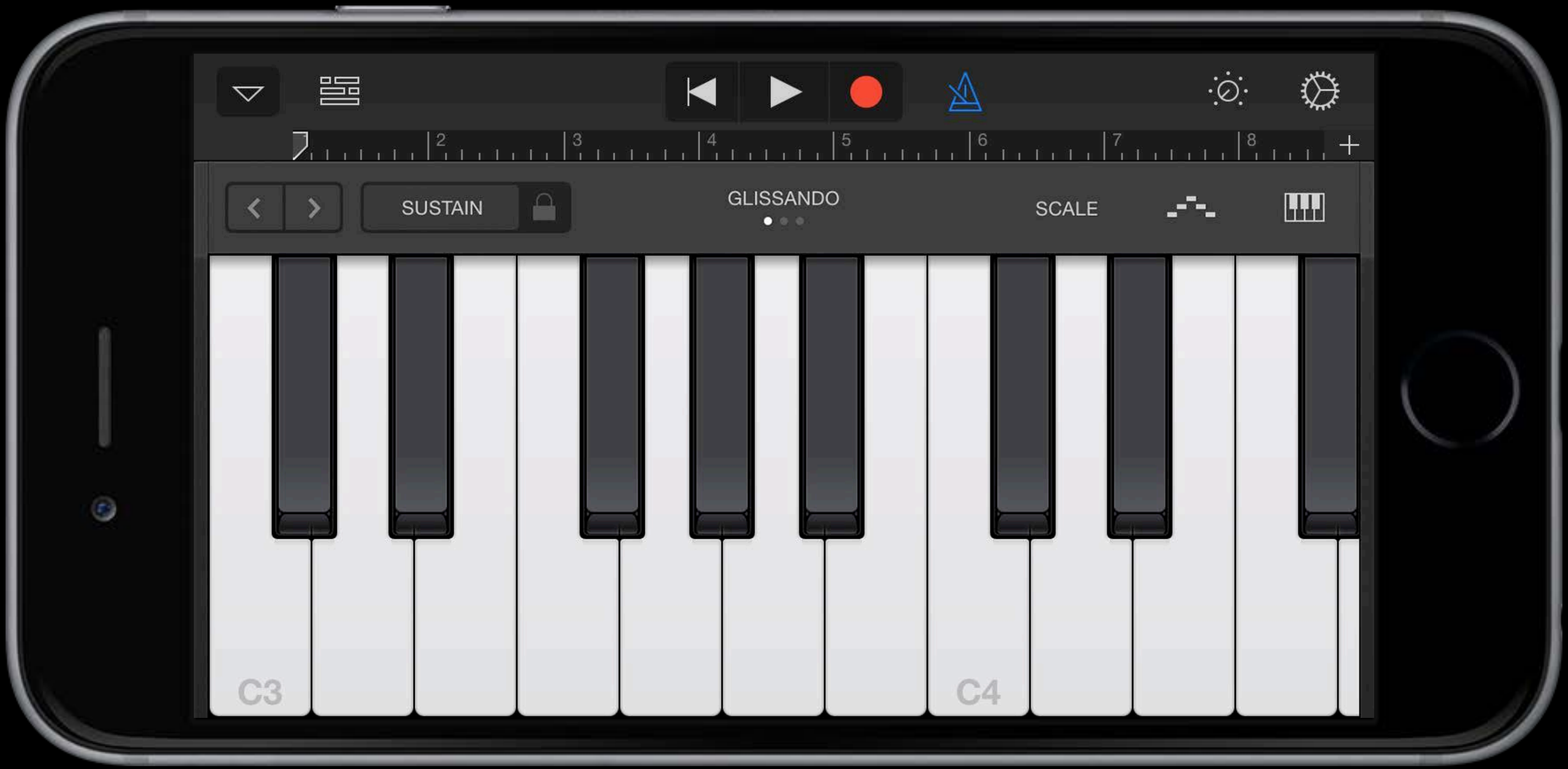
Audio Units

Lorem ipsum dolor sit amet, consectetur adipiscing elit.


iPad
9:41 AM
My Songs
Instruments
1
2
3
4
5
6
7
8
Custom View
0
SUSTAIN
GLISSANDO
SCALE
C2
C3
C4

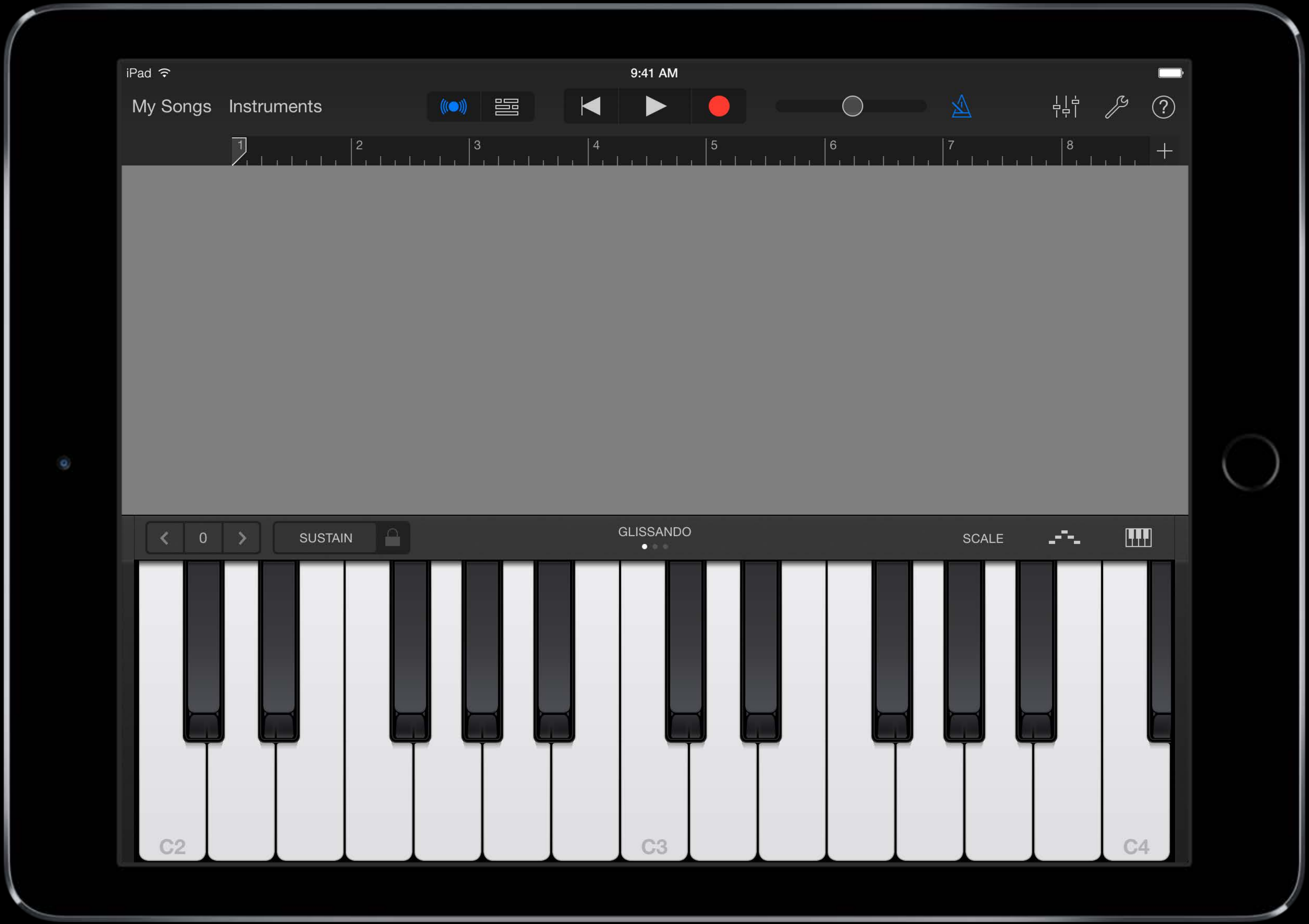

No redundant MIDI control surfaces here.

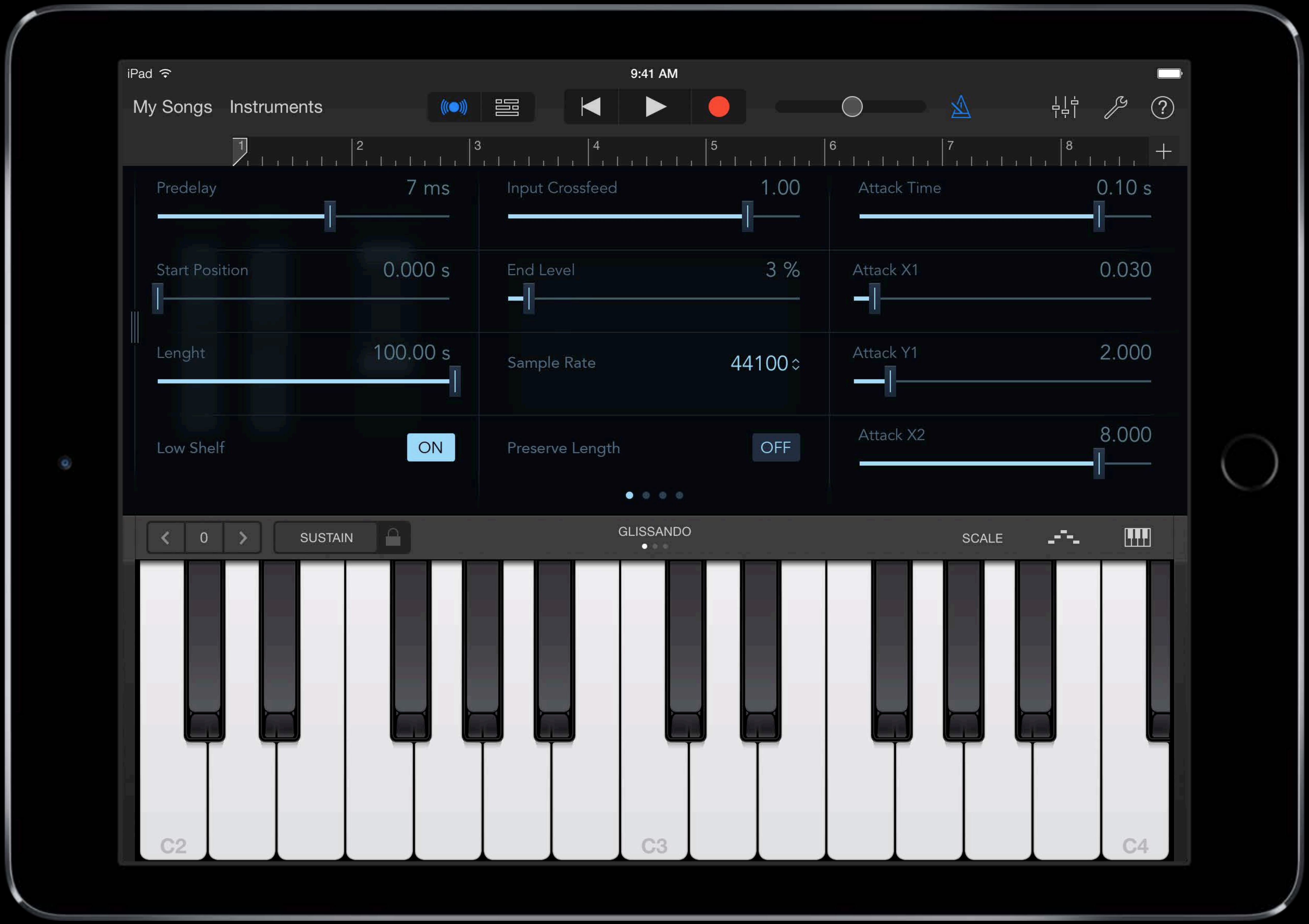

SUSTAIN
GLISSANDO
SCALE
C3
C4


SUSTAIN
GLISSANDO
SCALE
C3
C4

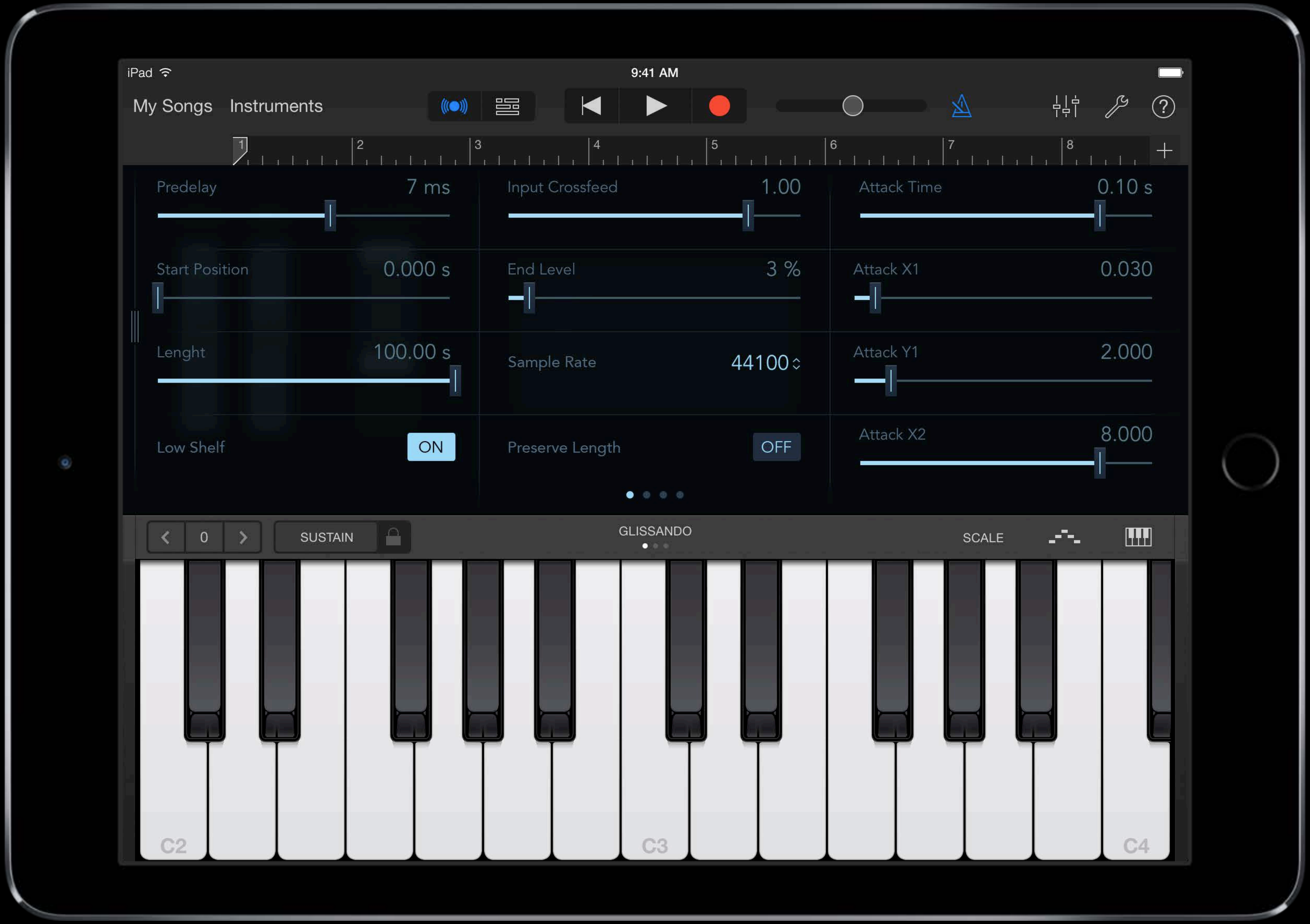
GarageBand iOS Custom View

iPad Air	2048 x 670
iPhone 6+	2208 x 726
iPhone 6	1334 x 404
iPhone 5s	1136 x 350


iPad
9:41 AM
My Songs
Instruments
1
2
3
4
5
6
7
8
0
SUSTAIN
GLISSANDO
SCALE
C2
C3
C4



Predelay 7 ms
Input Crossfeed 1.00
Attack Time 0.10 s
Start Position 0.000 s
End Level 3 %
Attack X1 0.030
Lenght 100.00 s
Sample Rate 44100
Attack Y1 2.000
Low Shelf ON
Preserve Length OFF
Attack X2 8.000



Predelay 7 ms
Start Position 0.000 s
Lenght 100.00 s
Low Shelf ON
Input Crossfeed 1.00
End Level 3 %
Sample Rate 44100
Preserve Length OFF
Attack Time 0.10 s
Attack X1 0.030
Attack Y1 2.000
Attack X2 8.000

Apple Music Creation Apps

Audio Unit Extensions

FAQ

What About Inter-App Audio?

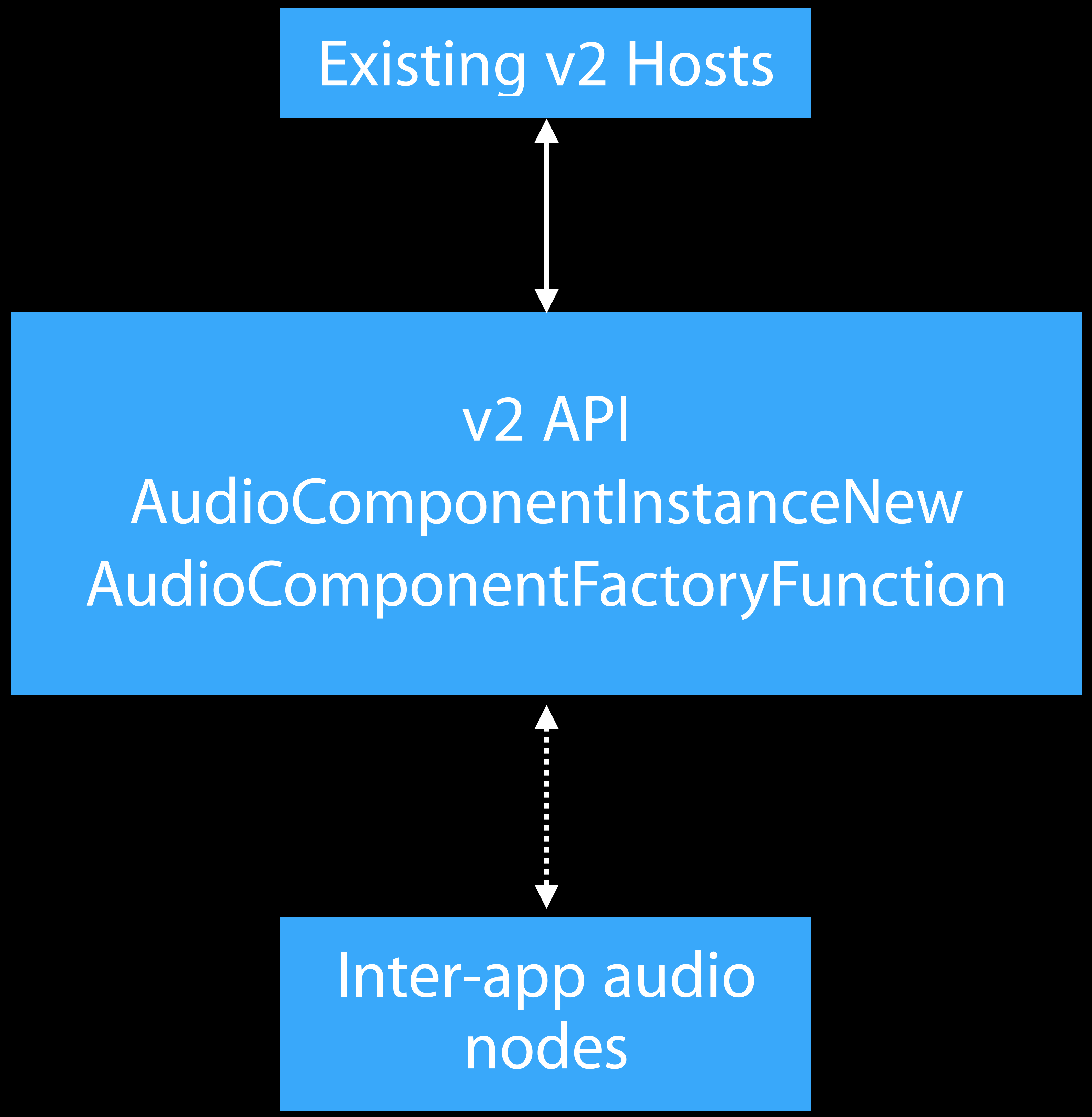
Inter-app audio uses a small subset of v2 API

No parameter support

Switching apps can be clunky

Not deprecated

Audio Unit Extensions—Inter-App Audio++



What About My v2 Host or AU?

Bridges maintain compatibility

Update to v3 when you can

v2 AU porting shortcut

- Subclass AUAudioUnitV2Bridge

Are v3 Audio Units Cross-Platform?

AUAudioUnit

- Fully portable

AUViewController

- Derives from UIViewController or NSViewController
- UI is platform-specific

Reference

Headers in AudioUnit framework—but link AudioToolbox

- AUAudioUnit.h
- AUAudioUnitImplementation.h
- AUParameters.h

CoreAudioKit framework

- AUViewController.h

AVFoundation

- AVAudioUnitComponent.h

HeaderDoc

Reference

License to use Audio Units logo

- https://developer.apple.com/softwarelicensing/agreements/audio.php

Summary

Full plug-in model on iOS

Audio Units in the iOS and OS X App Stores

`AVAudioEngine` simple host

Sample code: `AudioUnitV3Example`

Write bugs

More Information

Technical Support

Swift Language Documentation
http://developer.apple.com/swift

Apple Developer Forums
http://developer.apple.com/forums

Related Sessions

What's New in Core Audio	Nob Hill	Wednesday 4:30PM

Related Labs

Audio Lab	Graphics, Games, and Media Lab A	Thursday 1:30PM

WWDC15