CocosJuce

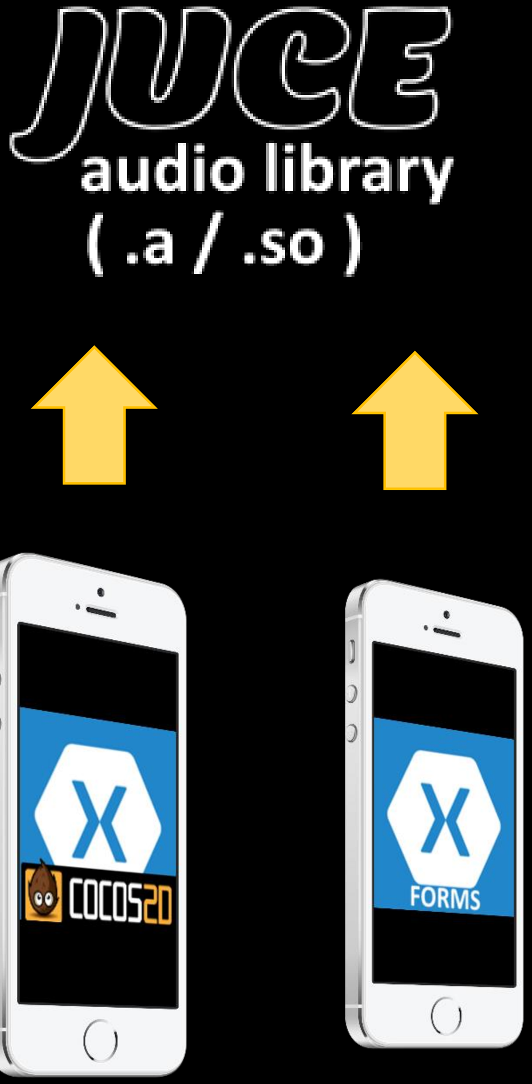
Combining Juce Audio with a Xamarin CocosSharp UI

Leo Olivers

Juce Summit Nov 2015

What’s this talk about?

- Build a headless Juce Audio library in C++
- Consume that library from a Xamarin CocosSharp game app
  - Integrate with a game UI
  - On iOS and Android
- Bonus: Consume same library from a Xamarin.Forms app
  - Integrate with a non-game UI
  - On iOS and Android



Build a headless Juce Audio Library : Design

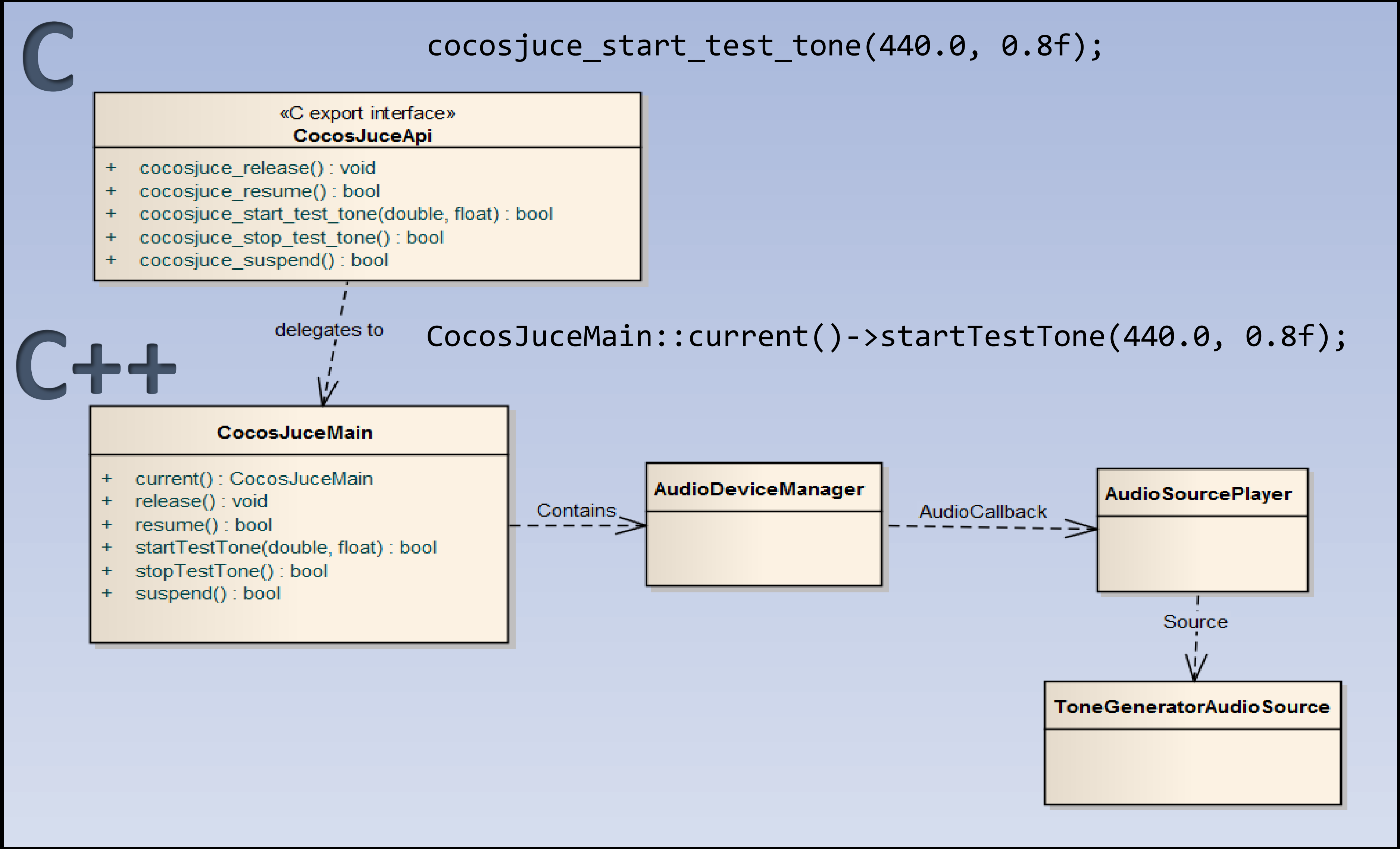
- Basic Example:

  AudioSourcePlayer
  +
  ToneGenerator-
  AudioSource

  No UI !

- Real World:

  you would probably host an AudioProcessor within an AudioProcessor-Player


C
C++
«C export interface»
CocosJuceApi
+ cocosjuce_release() : void
+ cocosjuce_resume() : bool
+ cocosjuce_start_test_tone(double, float) : bool
+ cocosjuce_stop_test_tone() : bool
+ cocosjuce_suspend() : bool
delegates to
CocosJuceMain
+ current() : CocosJuceMain
+ release() : void
+ resume() : bool
+ startTestTone(double, float) : bool
+ stopTestTone() : bool
+ suspend() : bool
Contains
AudioDeviceManager
AudioCallback
AudioSourcePlayer
Source
ToneGeneratorAudioSource

```
cocosjuce_start_test_tone(440.0, 0.8f);
```

```
CocosJuceMain::current()->startTestTone(440.0, 0.8f);
```

Build a headless Juce Audio Library: IntroJucer

- iOS

Introjucer Build Setting	Value
Library Type	Static Library (.a)
Additional Modules	juce_audio_utils
iOS Deployment Target	7.0
Keep defaults for all else	

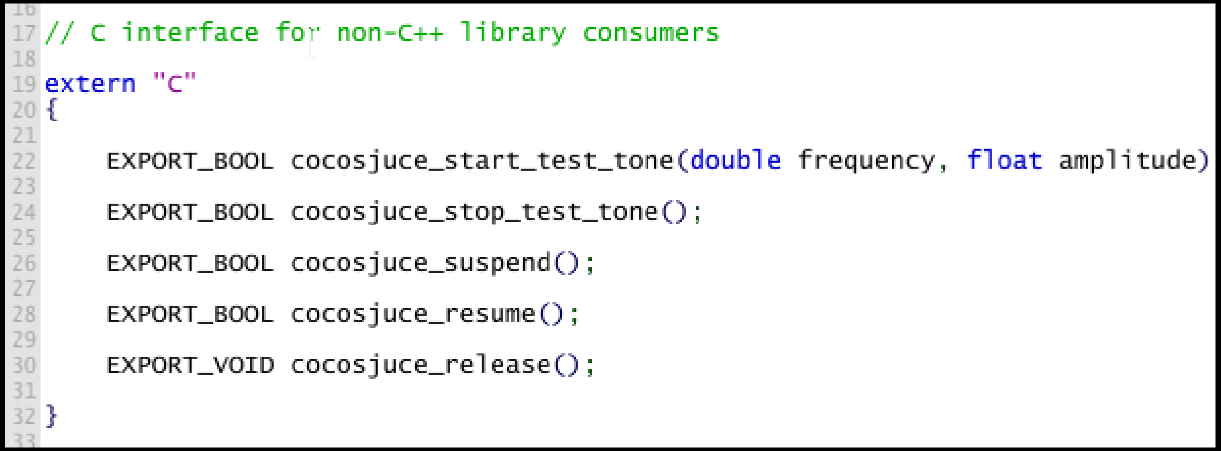
- Android

Introjucer Build Setting	Value
Library Type	Dynamic Library (.so)
Additional Modules	juce_audio_utils
Android Activity Class Name	com.yourcompany.cocosjucesharedlib.JuceActivity
Minimum SDK Version	16 (Android 4.1 Jelly Bean)
External Libraries to Link	android
Architectures	armeabi armeabi-v7a x86
Keep defaults for all else	

Tested with
Juce v3.3.0

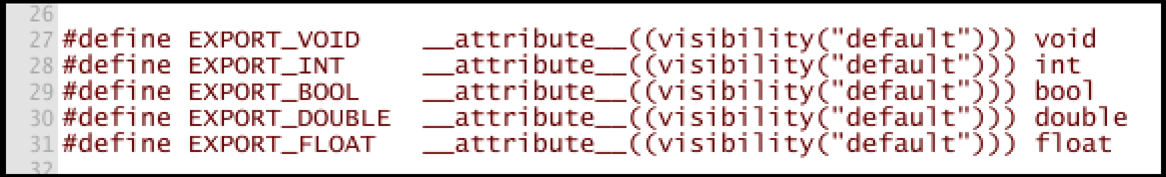
Build a headless Juce Audio Library: Exports

- C Exports require attention:



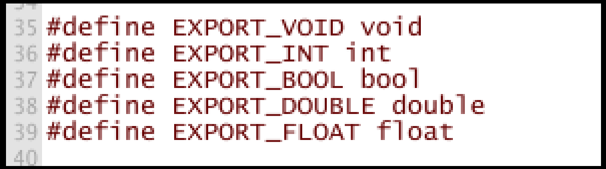
```
// C interface for non-C++ library consumers

extern "C"
{

    EXPORT_BOOL cocosjuce_start_test_tone(double frequency, float amplitude)

    EXPORT_BOOL cocosjuce_stop_test_tone();

    EXPORT_BOOL cocosjuce_suspend();

    EXPORT_BOOL cocosjuce_resume();

    EXPORT_VOID cocosjuce_release();

}
```

- iOS



```
#define EXPORT_VOID    __attribute__((visibility("default"))) void
#define EXPORT_INT     __attribute__((visibility("default"))) int
#define EXPORT_BOOL    __attribute__((visibility("default"))) bool
#define EXPORT_DOUBLE  __attribute__((visibility("default"))) double
#define EXPORT_FLOAT   __attribute__((visibility("default"))) float
```

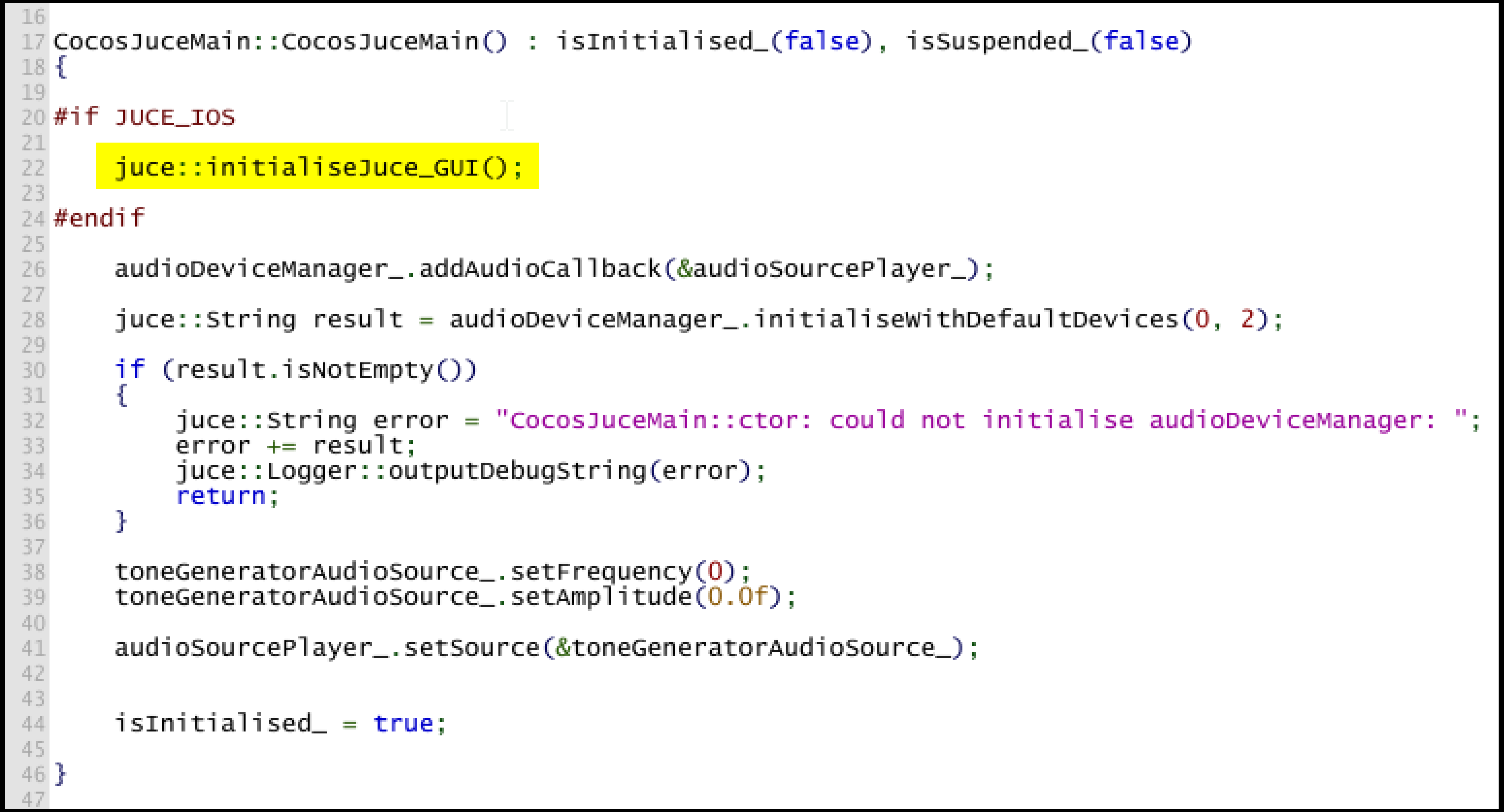
- Android



```
#define EXPORT_VOID void
#define EXPORT_INT int
#define EXPORT_BOOL bool
#define EXPORT_DOUBLE double
#define EXPORT_FLOAT float
```

Build a headless Juce Audio Library: Initialisation

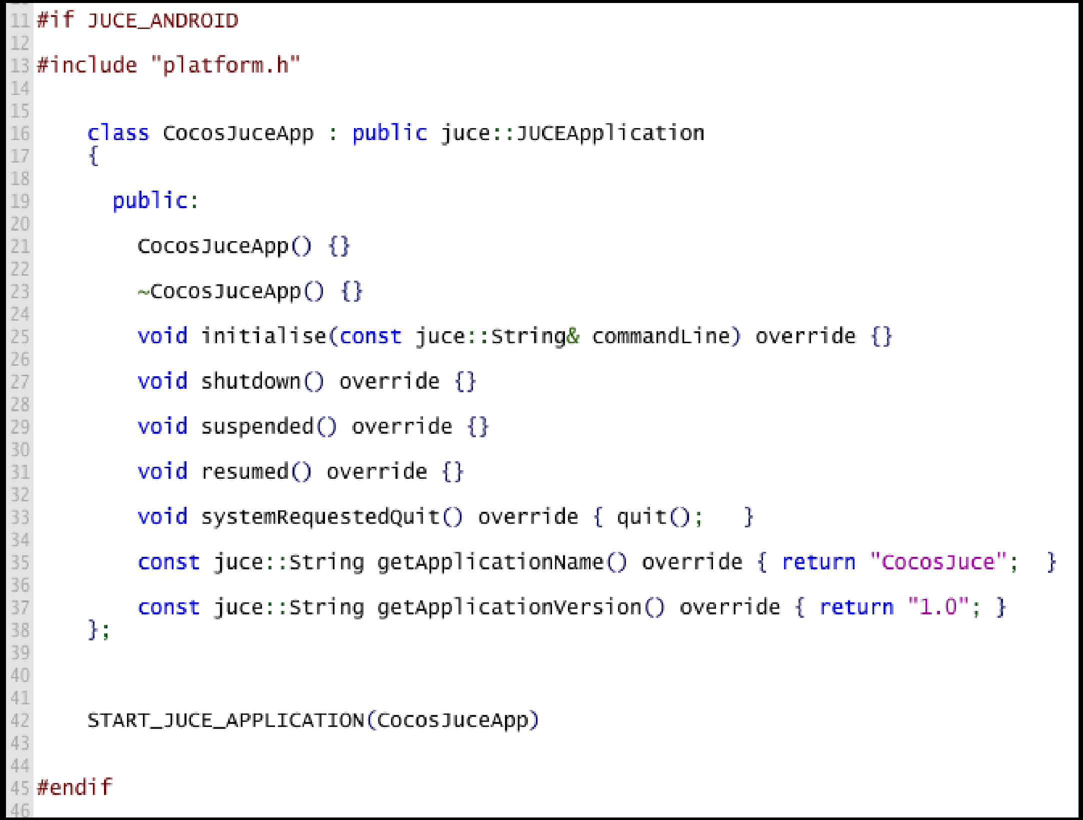
- On iOS, Juce initialisation is very straightforward
  - Just call juce::InitialiseJuce_GUI();



```
CocosJuceMain::CocosJuceMain() : isInitialised_(false), isSuspended_(false)
{

#if JUCE_IOS

    juce::initialiseJuce_GUI();

#endif

    audioDeviceManager_.addAudioCallback(&audioSourcePlayer_);

    juce::String result = audioDeviceManager_.initialiseWithDefaultDevices(0, 2);

    if (result.isNotEmpty())
    {
        juce::String error = "CocosJuceMain::ctor: could not initialise audioDeviceManager: ";
        error += result;
        juce::Logger::outputDebugString(error);
        return;
    }

    toneGeneratorAudioSource_.setFrequency(0);
    toneGeneratorAudioSource_.setAmplitude(0.0f);

    audioSourcePlayer_.setSource(&toneGeneratorAudioSource_);


    isInitialised_ = true;

}
```

Build a headless Juce Audio Library: Initialisation

- Android needs special JuceActivity.java class instead
  - Generated by IntroJucer
  - Provides Java/C++ interop
  - Initialises JUCE android system
  - We'll deal with it later on the Xamarin side
- Android also needs additional JUCEApplication class implementation
  - Must be there for linking to succeed. Does not play an important role
-



```
#if JUCE_ANDROID

#include "platform.h"


    class CocosJuceApp : public juce::JUCEApplication
    {

      public:

        CocosJuceApp() {}

        ~CocosJuceApp() {}

        void initialise(const juce::String& commandLine) override {}

        void shutdown() override {}

        void suspended() override {}

        void resumed() override {}

        void systemRequestedQuit() override { quit();   }

        const juce::String getApplicationName() override { return "CocosJuce";  }

        const juce::String getApplicationVersion() override { return "1.0"; }
    };



    START_JUCE_APPLICATION(CocosJuceApp)


#endif
```

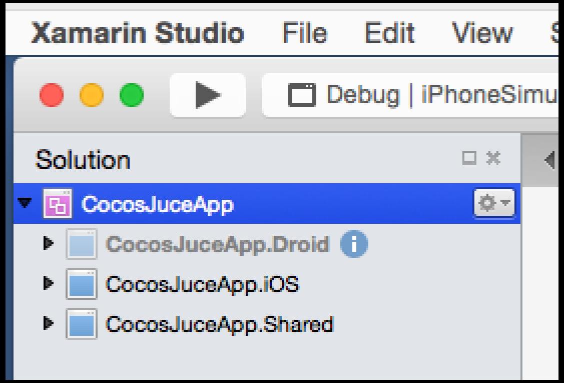
Build a headless Juce Audio Library: Building

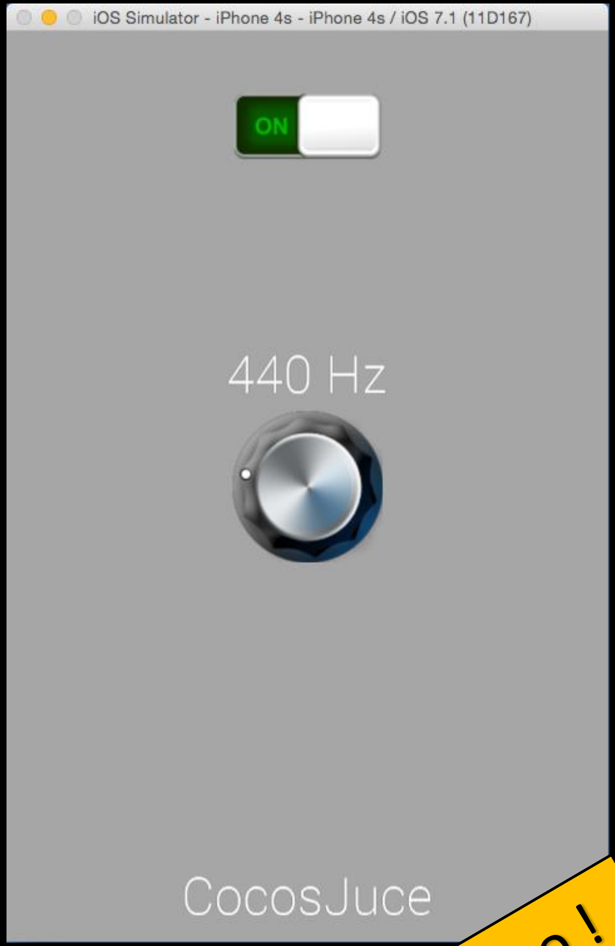
- iOS
  - Build 2 x with XCode 6.4   (not tested with XCode 7 but might work)
    - 1 x For iPhoneSimulator (build for Profiling)
    - 1 x For iPhone (build for Archiving)
    - Output: 2 x libCocosJuceStaticLib.a

- Android
  - Build with SDK 24.3.4 + API 16 / NDK r10e   (newer SDKs might work too)
    - “`cd CocosJuce/CocosJuceSharedLib/Builds/Android`”
    - “`ant release`”
    - Output:
      - Android/libs/armeabi/libjuce_jni.so
      - Android/libs/armeabi-v7a/libjuce_jni.so
      - Android/libs/x86/libjuce_jni.so
    - Rename these to 3 x libCocosJuceSharedLib.so

Consume Lib From CocosSharp Game : Projects

- About Xamarin and CocosSharp…
- Open CocosJuceApp.sln solution in Xamarin Studio or Visual Studio
- 3 Projects
  - CocosJuceApp.Droid. Contains Android startup code
  - CocosJuceApp.iOS. Contains iOS startup code
  - CocosJuceApp.Shared. Contains shared game code

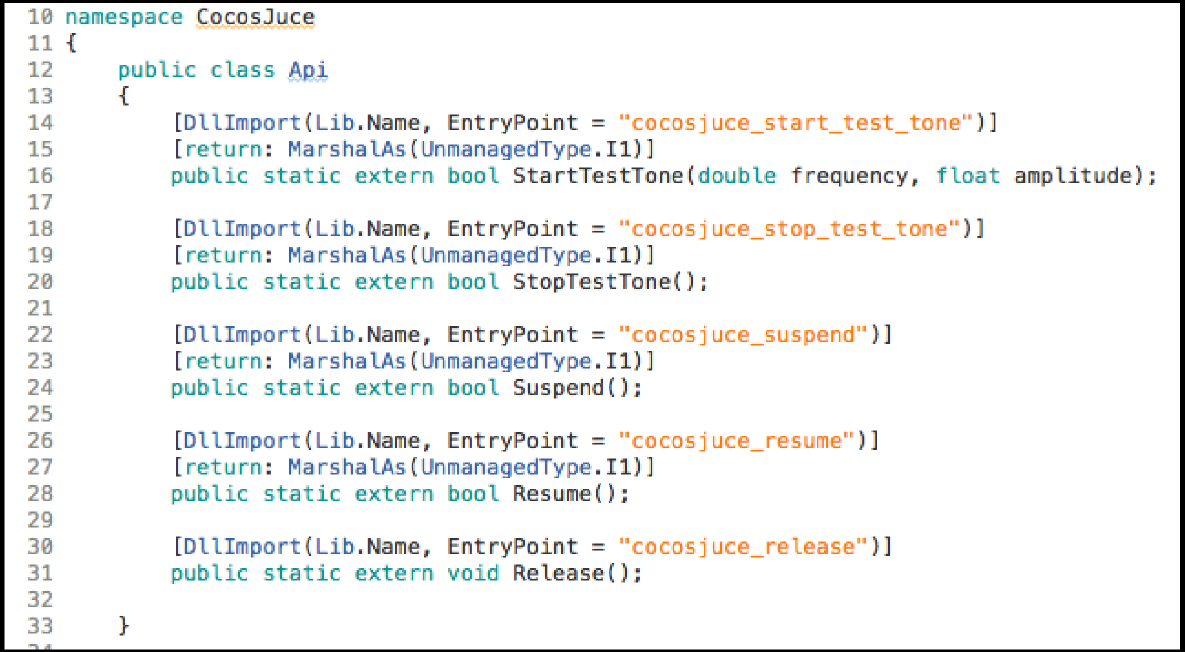




Demo !

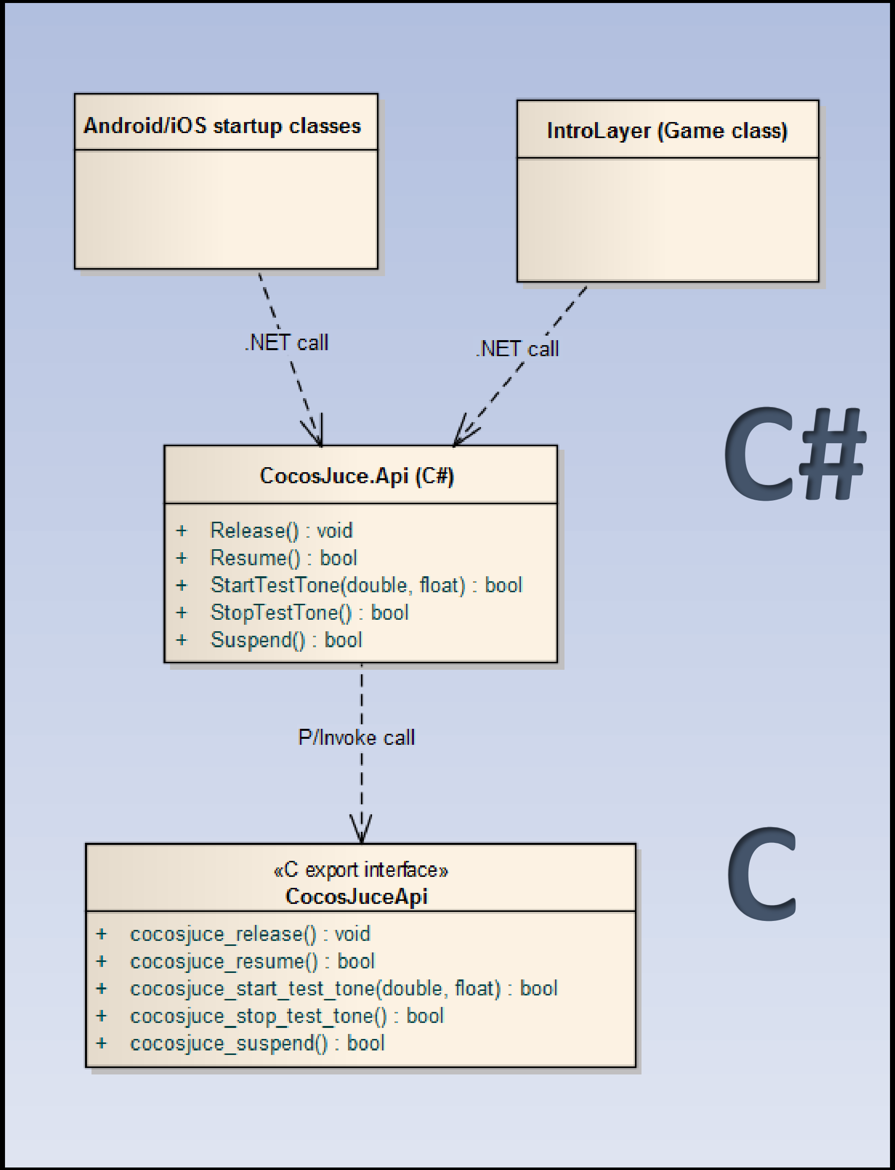
Consume Lib From CocosSharp Game: Design

- C# code is able to call C lib code via P/Invoke mechanism
- Defined in CocosJuceApi.cs



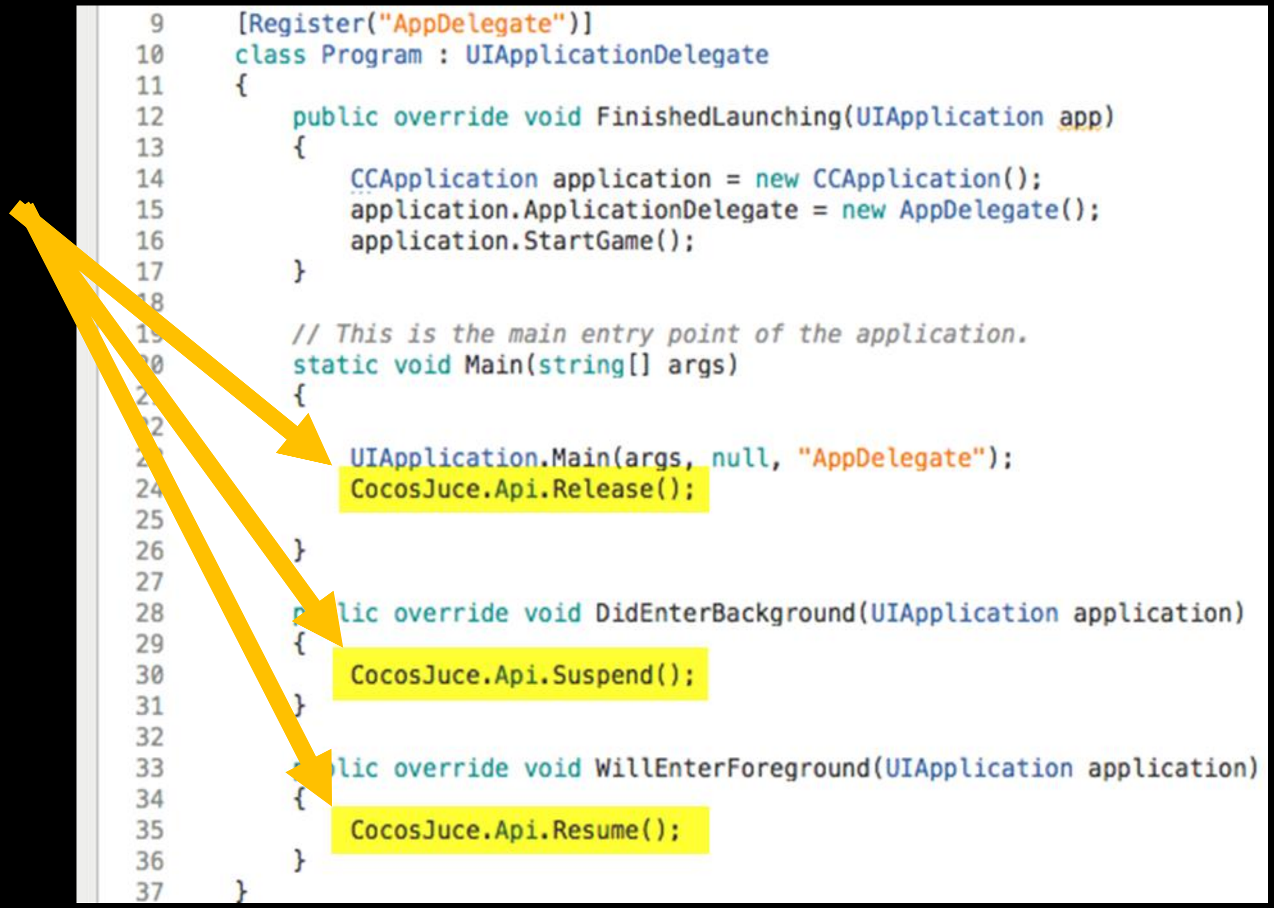
```
namespace CocosJuce
{
    public class Api
    {
        [DllImport(Lib.Name, EntryPoint = "cocosjuce_start_test_tone")]
        [return: MarshalAs(UnmanagedType.I1)]
        public static extern bool StartTestTone(double frequency, float amplitude);

        [DllImport(Lib.Name, EntryPoint = "cocosjuce_stop_test_tone")]
        [return: MarshalAs(UnmanagedType.I1)]
        public static extern bool StopTestTone();

        [DllImport(Lib.Name, EntryPoint = "cocosjuce_suspend")]
        [return: MarshalAs(UnmanagedType.I1)]
        public static extern bool Suspend();

        [DllImport(Lib.Name, EntryPoint = "cocosjuce_resume")]
        [return: MarshalAs(UnmanagedType.I1)]
        public static extern bool Resume();

        [DllImport(Lib.Name, EntryPoint = "cocosjuce_release")]
        public static extern void Release();

    }
```

- Need to include this class both in iOS and Android projects


Android/iOS startup classes
IntroLayer (Game class)
.NET call
.NET call
CocosJuce.Api (C#)
+ Release() : void
+ Resume() : bool
+ StartTestTone(double, float) : bool
+ StopTestTone() : bool
+ Suspend() : bool
C#
P/Invoke call
«C export interface»
CocosJuceApi
+ cocosjuce_release() : void
+ cocosjuce_resume() : bool
+ cocosjuce_start_test_tone(double, float) : bool
+ cocosjuce_stop_test_tone() : bool
+ cocosjuce_suspend() : bool
C

Consume Lib From CocosSharp Game: Coding

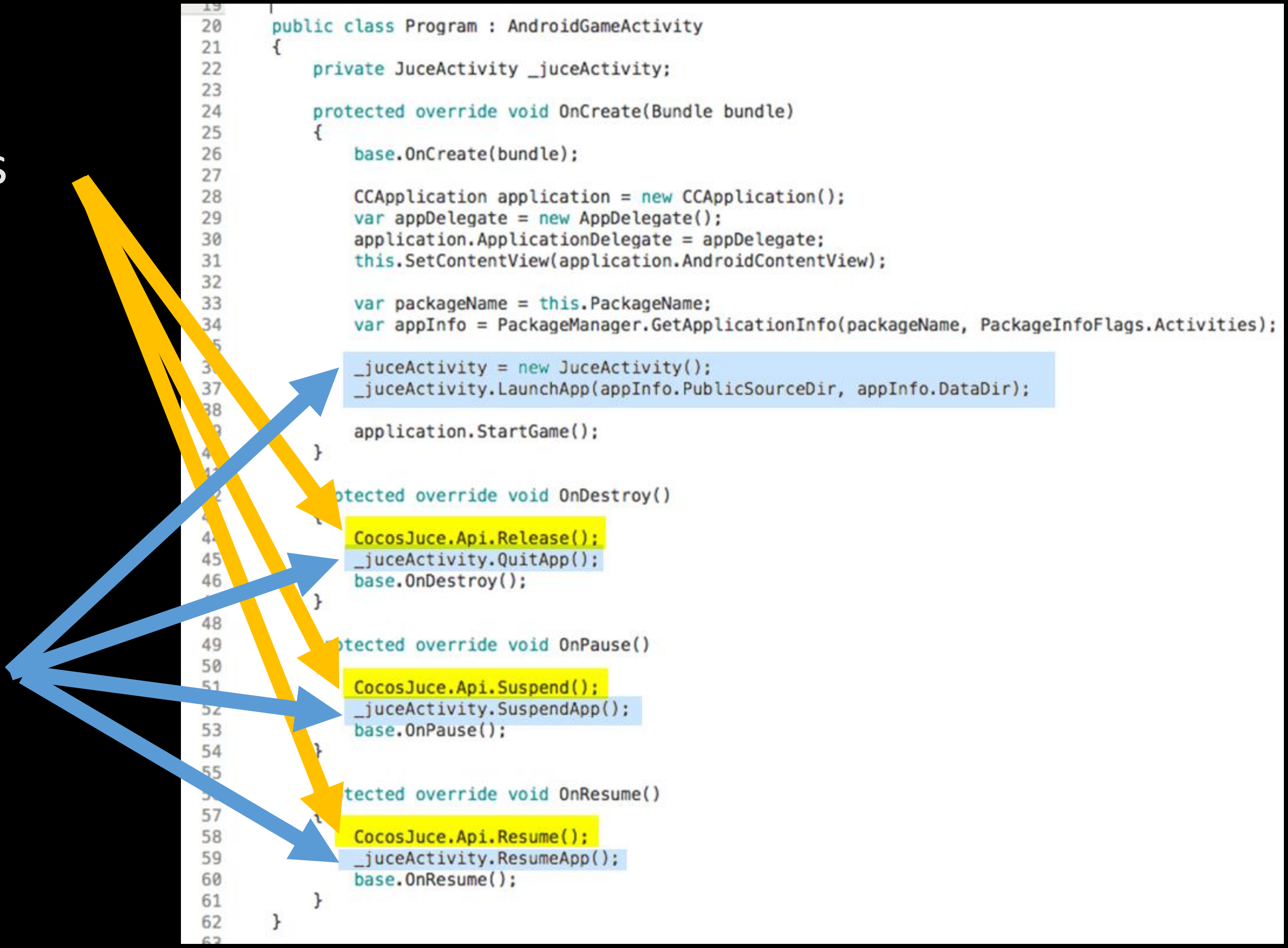
- iOS: In the AppDelegate class:
  - Call your lib's lifecycle methods at appropriate moments
  - No init method needed: Our lib's initialisation is automatic (lazy)



```
[Register("AppDelegate")]
class Program : UIApplicationDelegate
{
    public override void FinishedLaunching(UIApplication app)
    {
        CCApplication application = new CCApplication();
        application.ApplicationDelegate = new AppDelegate();
        application.StartGame();
    }

    // This is the main entry point of the application.
    static void Main(string[] args)
    {

        UIApplication.Main(args, null, "AppDelegate");
        CocosJuce.Api.Release();

    }

    public override void DidEnterBackground(UIApplication application)
    {
        CocosJuce.Api.Suspend();
    }

    public override void WillEnterForeground(UIApplication application)
    {
        CocosJuce.Api.Resume();
    }
}
```

Consume Lib From CocosSharp Game: Coding

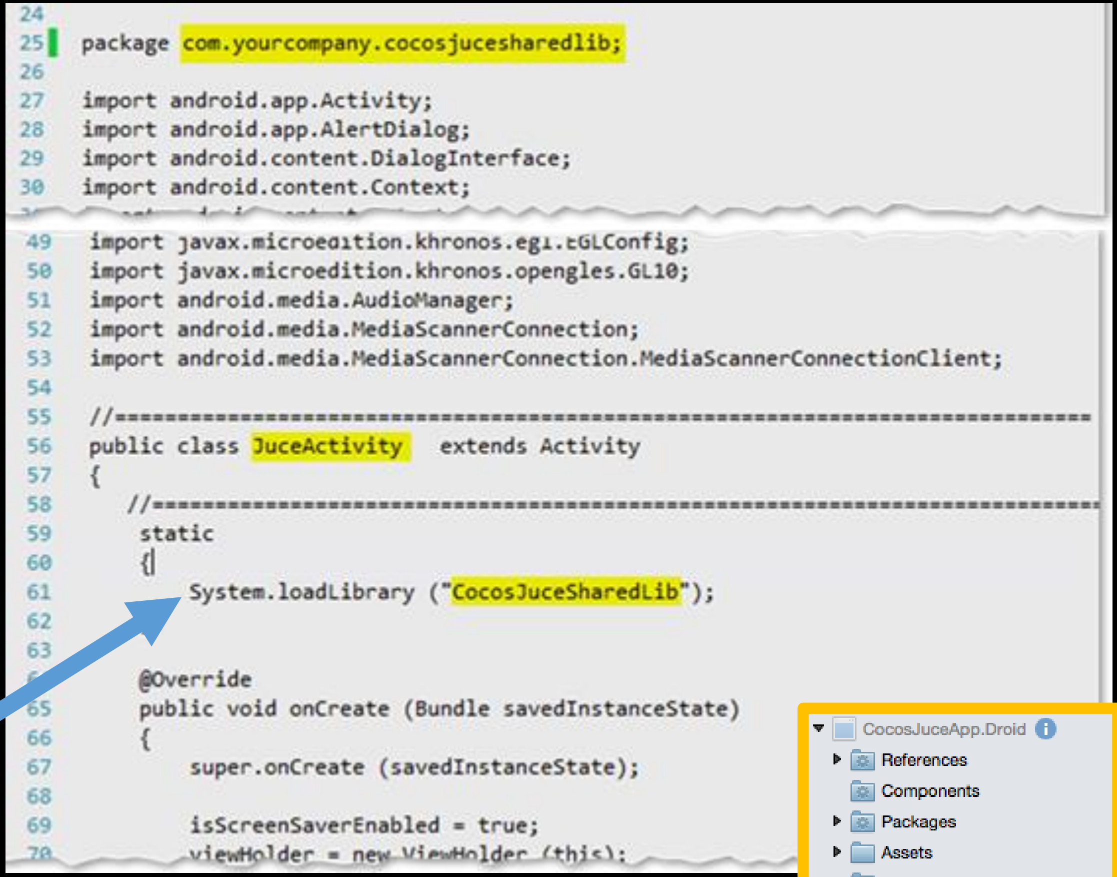
- Android: In the startup Activity class:
  - Call your lib's lifecycle methods at appropriate moments
  - No init method needed: Our lib's initialisation is automatic (lazy)
- Android needs an extra step: Manage the JuceActivity
  - Instantiate a JuceActivity class and call its lifecycle methods



```
    public class Program : AndroidGameActivity
    {
        private JuceActivity _juceActivity;

        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

            CCApplication application = new CCApplication();
            var appDelegate = new AppDelegate();
            application.ApplicationDelegate = appDelegate;
            this.SetContentView(application.AndroidContentView);

            var packageName = this.PackageName;
            var appInfo = PackageManager.GetApplicationInfo(packageName, PackageInfoFlags.Activities);

            _juceActivity = new JuceActivity();
            _juceActivity.LaunchApp(appInfo.PublicSourceDir, appInfo.DataDir);

            application.StartGame();
        }

        otected override void OnDestroy()
        {
            CocosJuce.Api.Release();
            _juceActivity.QuitApp();
            base.OnDestroy();
        }

        otected override void OnPause()

            CocosJuce.Api.Suspend();
            _juceActivity.SuspendApp();
            base.OnPause();
        }

         tected override void OnResume()
        {
            CocosJuce.Api.Resume();
            _juceActivity.ResumeApp();
            base.OnResume();
        }
    }
```

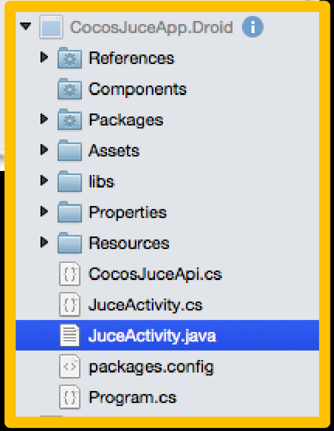
Consume Lib From CocosSharp Game: Coding

- JuceActivity.java
  - Is the original Android startup activity class that IntroJucer generated for us
    - Look for it in CocosJuce/CocosJuceSharedLib/Builds/Android/src/com/yourcompany/cocosjucesharedlib/JuceActivity.java
  - We cannot let it be the startup activity though -> that needs to be derived from CocosSharp AndroidGameActivity
  - So instead, we delegate to it from within our own startup activity
  - But we need to fix this loadLibrary call first
- JuceActivity.cs
  - Is a C# wrapper/interop class for JuceActivity.java
  - Used by C# code to instantiate and call JuceActivity.java methods



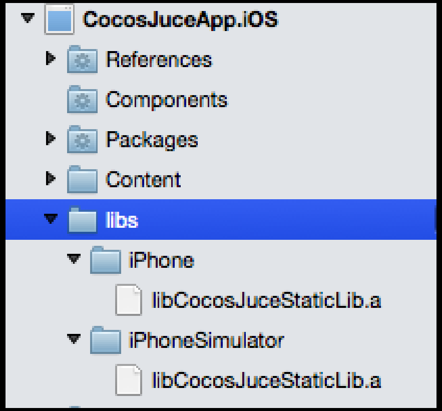
```

package com.yourcompany.cocosjucesharedlib;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Context;
```

```
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import android.media.AudioManager;
import android.media.MediaScannerConnection;
import android.media.MediaScannerConnection.MediaScannerConnectionClient;

//==============================================================================
public class JuceActivity   extends Activity
{
    //==============================================================================
    static
    {
        System.loadLibrary ("CocosJuceSharedLib");
    }

    @Override
    public void onCreate (Bundle savedInstanceState)
    {
        super.onCreate (savedInstanceState);

        isScreenSaverEnabled = true;
        viewHolder = new ViewHolder (this);
```



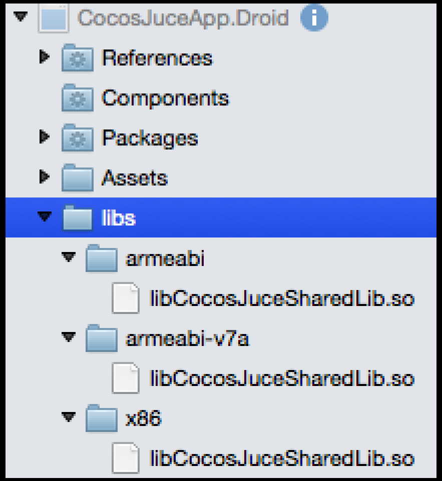
Consume Lib From CocosSharp Game: Linking

- Xamarin can link in native libs. For this you have to add them to the solution
- iOS:
  - Build Action: None
  - Copy To Output Dir: Always copy



```
CocosJuceApp.iOS
  References
  Components
  Packages
  Content
  libs
    iPhone
      libCocosJuceStaticLib.a
    iPhoneSimulator
      libCocosJuceStaticLib.a
```

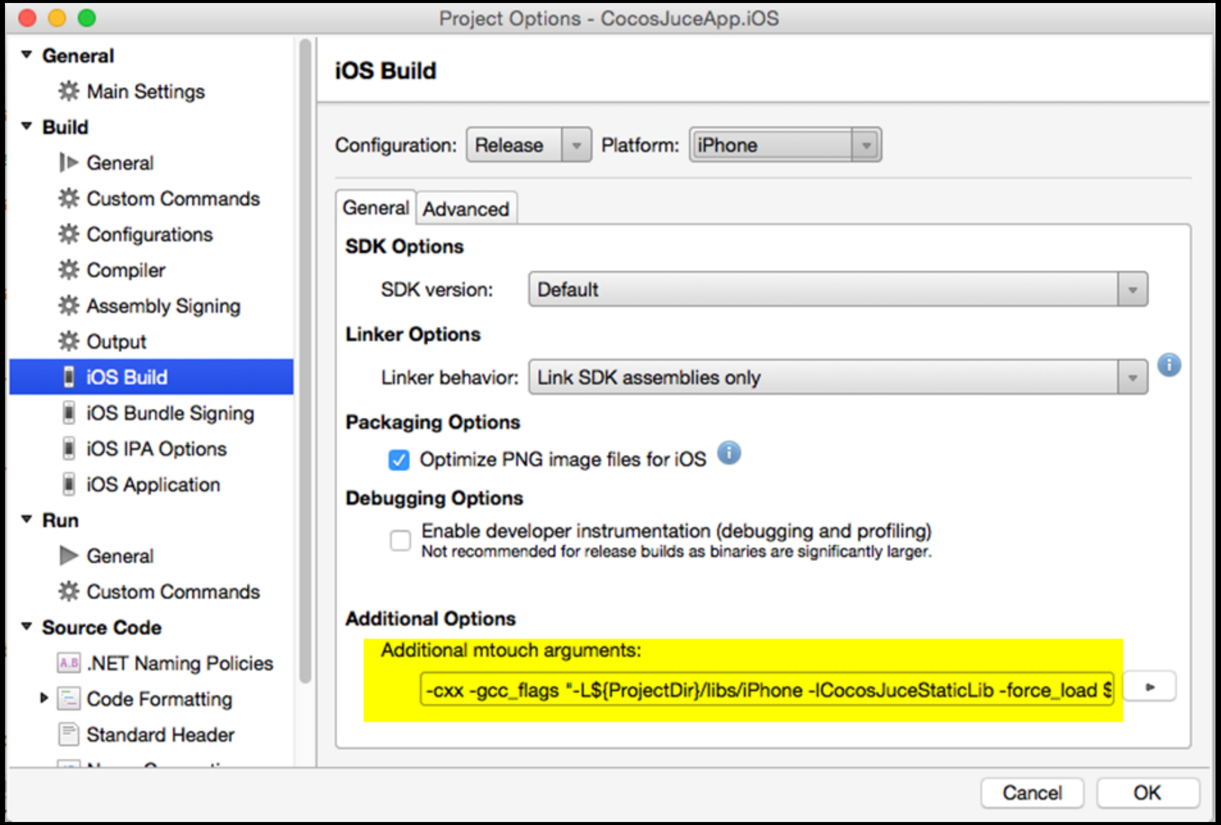
- Android
  - Build Action: AndroidNativeLibrary
  - Copy To Output Dir: Do not copy
  - Mandatory folder structure!



```
CocosJuceApp.Droid
  References
  Components
  Packages
  Assets
  libs
    armeabi
      libCocosJuceSharedLib.so
    armeabi-v7a
      libCocosJuceSharedLib.so
    x86
      libCocosJuceSharedLib.so
```

Consume Lib From CocosSharp Game: Linking

- Android library linking happens automatically
- iOS requires extra project-level settings ("mtouch arguments")


Project Options - CocosJuceApp.iOS

- General
  - Main Settings
- Build
  - General
  - Custom Commands
  - Configurations
  - Compiler
  - Assembly Signing
  - Output
  - iOS Build
  - iOS Bundle Signing
  - iOS IPA Options
  - iOS Application
- Run
  - General
  - Custom Commands
- Source Code
  - .NET Naming Policies
  - Code Formatting
  - Standard Header

iOS Build

Configuration: Release Platform: iPhone

General | Advanced

SDK Options

SDK version: Default

Linker Options

Linker behavior: Link SDK assemblies only

Packaging Options

[✓] Optimize PNG image files for iOS

Debugging Options

[ ] Enable developer instrumentation (debugging and profiling)
Not recommended for release builds as binaries are significantly larger.

Additional Options

Additional mtouch arguments:

```
-cxx -gcc_flags "-L${ProjectDir}/libs/iPhone -lCocosJuceStaticLib -force_load $
```

Cancel OK

Consume Lib From CocosSharp Game: Linking

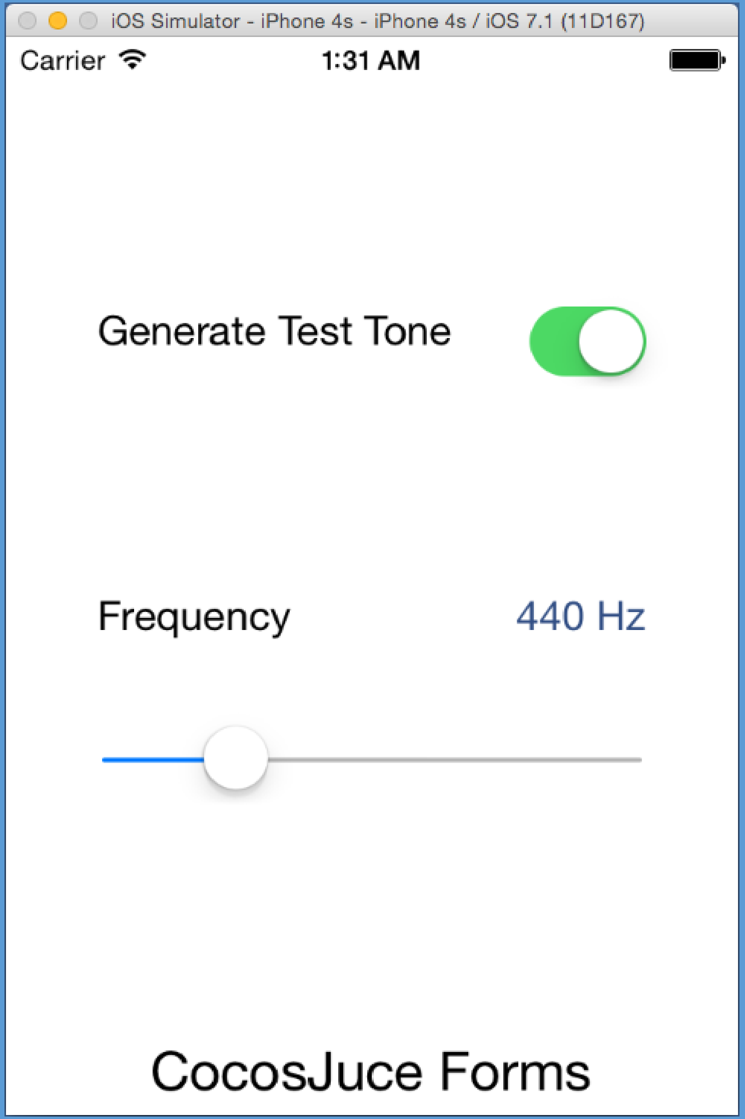
Build configuration	Additional mtouch arguments
iPhoneSimulator Debug/Release	-cxx -gcc_flags "-L${ProjectDir}/libs/iPhoneSimulator -lCocosJuceStaticLib -force_load ${ProjectDir}/libs/iPhoneSimulator/libCocosJuceStaticLib.a -framework CoreText -framework AudioToolbox -framework CoreMidi -framework Accelerate"
iPhone Debug/Release	-cxx -gcc_flags "-L${ProjectDir}/libs/iPhone -lCocosJuceStaticLib -force_load ${ProjectDir}/libs/iPhone/libCocosJuceStaticLib.a -framework CoreText -framework AudioToolbox -framework CoreMidi -framework Accelerate"

Bonus: Consume Lib From Non-Game UI

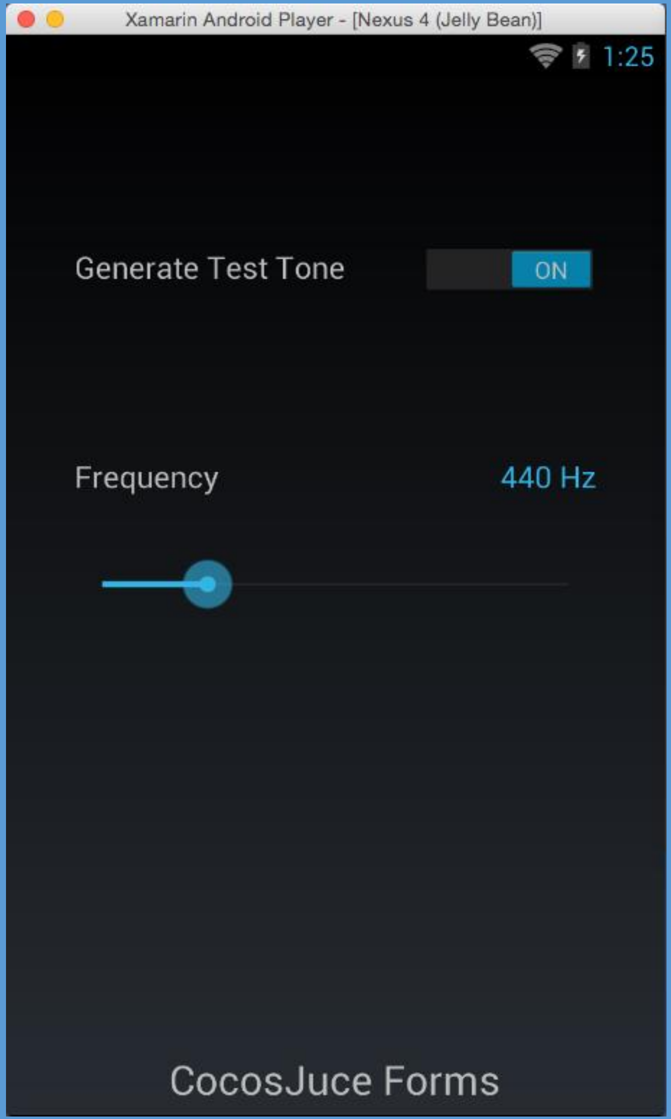
- **Provided in sample solution CocosJuceFormsApp.sln**
- **UI built with Xamarin.Forms = x-platform API for mobile UI**
- **Native look & feel**
- **Alternative: could also build the UIs separately using traditional native iOS / Android APIs**
- **Juce lib binding : identical to CocosSharp version**

Bonus: Consume Lib From Non-Game UI

iOS



Android



Wrapping up

- **Sample source code available at**
  github.com/altalogix/cocosjuce

- **Blog article available at**
  www.mucoder.net/blog

- **Contact me in case of questions**
  leo.olivers@altalogix.com

Q & A