DiaDrawScreenCast ANE

Manual

Contents

What is DiaDrawScreenCast.ane?

What does it do?

DiaDrawScreenCast.ane is an AIR Native Extension for iOS which allows you to make a video recording of a given DisplayObject in your app and mix in pre-recorded sound. You have the option of encoding the videos with H.256 or JPEG and selecting video quality. See VideoSettings for full details.

How do you use DiaDrawScreenCast.ane?

Include **DiaDrawScreenCast.ane** in your Flash or Flex project.

Make an instance of the ScreenRecorder class (com.diadraw.extensions.ScreenRecorder) and use that to start, pause, resume or stop a recording.

Before you start a recording, make an instance of the VideoSettings class (com.diadraw.extensions.VideoSettings) and leave it set up by default or modify how your video should be recorded. VideoSettings lets you set:

- video frame size
- scale factor and scale mode (fill/fit/crop) for the DisplayObject in the video
- frame rate of the recorded video
- average video bit rate
- type of video encoding
- video encoding quality
- the path to an audio file, if you want to mix sound; and the following audio settings:
  - audio sample rate
  - audio bit rate
  - number of audio channels
  - video composition quality of the mixed video
  - offset in seconds, if you need the sound to start at a certain point in the video
  - audio looping and cut-off

Choose a name and path for your video file and call startRecording() on your ScreenRecorder instance. When you are ready to finish recording, decide whether you would also like a copy saved to the device's Camera Roll and call finishRecording().

Listen for ScreenRecorderEvent.VIDEO_SAVED to get notified when the video file has been written successfully.

Listen for ScreenRecorderEvent.ERROR to get notified about any issues during the recording or saving.

Example Code

Instantiate ScreenRecorder

```
import com.diadraw.extensions.ScreenRecorder;
import com.diadraw.extensions.ScreenRecorderEvent;
import com.diadraw.extensions.VideoSettings;

private function initializeScreenRecorder() : void
{
      m_screenRecorder = new ScreenRecorder();
      m_screenRecorder.addEventListener(
                  ScreenRecorderEvent.VIDEO_SAVED, onVideoSaved );
      m_screenRecorder.addEventListener(
                  ScreenRecorderEvent.ERROR, onScreenCastError );
      m_screenRecorder.addEventListener(
                  ScreenRecorderEvent.PROGRESS, onVideoSaveProgress );
}
```

Start recording a video

```
private function startRecording() : void
{
      // 1. Get the container that will be recorded
      var containerToRecord : DisplayObject = groupToRecord;

      // 2. ... and the optimal video size,
      // based on the container size and the video codec
      // When using VideoCodecH264 the video width must be
      // one of the standard video widths: 320, 640, etc.
      // When using VideoCodecJpeg both the width and the height of the video must be
      //  one of the standard ones.
      var videoCodec : Number = VideoSettings.VideoCodecJpeg; //VideoCodecH264;
      var videoSize : Point =
            VideoSettings.getStandardVideoSize(
                  containerToRecord.width,
                  containerToRecord.height,
                  videoCodec );

      // 3. If the video size isn't the same as the container's,
      // decide whether we want the container centered in the video frame
      // (or offset in any way)
      var containerOffsetInFrame : Point = new Point();

      // 4. Configure the rest of the video settings
      var videoSettings : VideoSettings = new VideoSettings();

      videoSettings.videoCodec = videoCodec; // Can be VideoSettings.VideoCodecH264
                                             // or VideoCodecJpeg.
      videoSettings.jpegQuality = 0.5; // VideoSettings.VideoCodecJpeg only.
                                       // Takes values between 0.0 and 1.0.
                                       // Specifies JPEG coded quality.
```

```
            videoSettings.h264Quality = VideoSettings.H264Main32;
                                        // VideoSettings.VideoCodec264 only
                                        // Takes the following values:
                                        // VideoSettings.H264Baseline30
                                        // VideoSettings.H264Baseline31
                                        // VideoSettings.H264Baseline41
                                        // VideoSettings.H264BaselineAuto
                                                // Supported on iOS > 7, defaults to
                                                // VideoSettings.H264Baseline30
                                                // on earlier iOS versions
                                        // VideoSettings.H264Main30
                                        // VideoSettings.H264Main31
                                        // VideoSettings.H264Main32
                                        // VideoSettings.H264Main41
                                        // VideoSettings.H264MainAuto
                                                // Supported on iOS > 7, defaults to
                                                // VideoSettings.H264Baseline30
                                                // on earlier iOS versions
                                        // VideoSettings.H264High40
                                                // Supported on iOS > 6, defaults to
                                                // VideoSettings.H264Main31
                                                // on earlier iOS versions
                                        // VideoSettings.H264High41
                                                // Supported on iOS > 6, defaults to
                                                // VideoSettings.H264Main31
                                                // on earlier iOS versions
                                        // VideoSettings.H264HighAuto
                                                // Supported on iOS > 7, defaults to
                                                // VideoSettings.H264Main31
                                                // on earlier iOS versions

        videoSettings.framesPerSecond = 15;             // Frames per second
        videoSettings.videoAverageBitRate = 960000;     // H264 only, bits per second

        // This sets the quality preset to be used
        // if the video is mixed with sound after it's been recorded
        // Use one of the following:
                // VideoSettings.VideoCompositionPresetLowQuality
                // VideoSettings.VideoCompositionPresetMediumQuality
                // VideoSettings.VideoCompositionPresetHighestQuality
        videoSettings.videoCompositionQuality =
                            VideoSettings.VideoCompositionPresetHighestQuality;

        // If the recorded container and the video frame are different sizes,
        // videoSettings.scale mode can set scaling, so that the container
        // either fills the video frame or fits in it.
        // Use VIDEO_SCALE_NONE, VIDEO_SCALE_FIT_W, VIDEO_SCALE_FIT_H
        // or VIDEO_SCALE_FIT_ALL.
        videoSettings.scaleMode = VideoSettings.VIDEO_SCALE_NONE;

        // scaleFactor allows you to scale the container with respect to the video frame
        // and takes x and y factors.
        // Note: scaleFactor takes priority over scaleMode.
        // Set scaleFactor to (1, 1) for no scaling or for scaleMode to have effect.
        videoSettings.scaleFactor = new Point( 1, 1 );


        // Optional audio settings
        var inputAudioFile : File =
                File.applicationDirectory.resolvePath( "assets/exam_piece.m4a" );
```

```
        videoSettings.inputAudioFilePath = inputAudioFile.nativePath;
        // Set inputAudioFilePath = null to disable adding sound
        videoSettings.audioOffsetSeconds = 0;
        videoSettings.loopAudio = false;
        videoSettings.cutOffAudioAtEndOfVideo = true;


        // Set saveDebugFrames to true to have every 10th frame
        // saved to CameraRoll - for debugging purposes only
        var saveDebugFrames : Boolean = false;

        if ( m_screenRecorder.startRecording( containerToRecord,
                                              videoSize.x,
                                              videoSize.y,
                                              containerOffsetInFrame,
                                              videoSettings,
                                              saveDebugFrames ) )
        {
                m_isRecording = true;

                m_hasRecordedVideo = false;
                title = "Recording in progress...";
        }
        else
        {
                title = "ERROR: Could not start recording";
        }
}
```

Pause a recording

```
private function pauseRecording() : void
{
        m_screenRecorder.pauseRecording();
        title = "Recording paused";

        btnPause.label = "Restart";
}
```

Resume a paused recording

```
private function pauseRecording() : void
{
        m_screenRecorder.pauseRecording();
        title = "Recording paused";

        btnPause.label = "Restart";
}
```

Finish recording and save video

```
private function finishRecording() : void
{
        // Note: You can save the file to a path of your choice and/or to Camera Roll.
        // You can also choose not to save it anywhere -
        // a bit pointless, but it's an option.

        // 1. If you want the file saved to a specific path,
        // pass that path as the second argument
        // in the m_screenRecorder.finishRecording() call.
```

```
        // If you don't want to save the file to a specific path,
        // leave that second argument out:
        // m_screenRecorder.finishRecording( true )
        // will only save a copy to Camera Roll.

        // 1.1. If saving the file to a path of your choice,
        // make sure any subfolders on that path already exist:
        var subDir : File = File.documentsDirectory.resolvePath( "diadraw/videos" );
        if ( !subDir.exists )
        {
                subDir.createDirectory();
        }

        // 1.2. Get the full path to the file, including the file name.
        var videoFile : File =
                File.documentsDirectory.resolvePath( "diadraw/videos/screenCast.mp4" );
        var saveVideoToPath : String = videoFile.nativePath;

        // 2. Do you want a copy in Camera Roll?
        var saveVideoToCameraRoll : Boolean = true;

        // 3. Now call finishRecording() in the ANE:
        m_screenRecorder.finishRecording( saveVideoToCameraRoll, saveVideoToPath );

        m_isRecording = false;
        title = "Saving video file...";
}
```

Get notified about progress, when the video is being saved

```
private function onVideoSaveProgress( _event : ScreenRecorderEvent ) : void
{
        title = "Encoding in progress... " + _event.progressPercent + "%";
}
```

Get notified when the video file has been saved

```
private function onVideoSaved( _event : ScreenRecorderEvent ) : void
{
        title = "Video saved";
        m_hasRecordedVideo = true;
}
```

Process errors

```
private function onScreenCastError( _event : ScreenRecorderEvent ) : void
{
        // deal with errors from the native extension
        trace( "ScreenCastNativeExtension error: " + _event.message );
}
```

API

ScreenRecorder class
Package com.diadraw.extensions

`ScreenRecorder()` constructor
public function ScreenRecorder(_target : IEventDispatcher = **null**)

Creates a new instance of ScreenRecorder.

`isSupported()` method
public function isSupported() : Boolean

Checks if screen recording is supported on the current platform.

`startRecording()` method
public function startRecording(_objectToRecord   : DisplayObject,
                                      _videoWidth   : int,
                                      _videoHeight   : int,
                                      _objectOffsetInFrame : Point,
                                      _videoSettings   : VideoSettings,
                                      _saveDebugFrames   : Boolean = **false**) : Boolean

Starts the video recording.

Parameters
_objectToRecord - a DisplayObject that will be recorded in the video
_videoWidth - the width of the final video
_videoHeight - the height of the final video
_objectOffsetInFrame - (x, y) offset for the DisplayObject in the video frame
_videoSettings, see com.diadraw.extensions.VideoSettings
_saveDebugFrames - optional, defaults to false; set to true to have every 10th frame output to Camera Roll for inspection

Returns
true if the recording started successfully, **false** otherwise.

See also
VideoSettings.getStandardVideoSize() for picking an optimal video size for your chosen encoder.

`restartRecording()` method
public function restartRecording() : void

Resumes a paused video recording. Listen for ScreenRecorderEvent.RECORDING_RESUMED to get notified when the video recording has started.

pauseRecording() method

public function pauseRecording() : **void**

Pauses the video recording. Listen for ScreenRecorderEvent.RECORDING_PAUSED to get notified when the video recording has paused successfully.

finishRecording() method

public function finishRecording(_shouldSaveToCameraRoll : Boolean = **false**,
                                     _saveFileToPath   : String = **null**) : **void**

Stops the recording and saves the video to a file. Listen for ScreenRecorderEvent.VIDEO_SAVED to get notified when the video file has been written successfully.

Parameters

_shouldSaveToCameraRoll - optional; set to true to have the video saved to Camera Roll, as well as to the path specified in startRecording();

_saveFileToPath - optional; set to the path where you would like the video file to be stored (other than in camera roll).

ScreenRecorderEvent class

package com.diadraw.extensions

ScreenRecorderEvent.VIDEO_SAVED event type
Emitted when the video file has finished saving.

ScreenRecorderEvent.RECORDING_PAUSED event type

Emitted when the recording of the video has been paused.

ScreenRecorderEvent.RECORDING_RESUMED event type
Emitted when a paused video recording has been resumed.

ScreenRecorderEvent.ERROR event type
Emitted when there has been a problem with the video recording. Check the ScreenRecorderEvent **message** property for details on the problem.

ScreenRecorderEvent.INFO event type
Listen to this event to get information at various points of the recording. Check the ScreenRecorderEvent **message** property for details.

ScreenRecorderEvent.PROGRESS event type
Listen to this event to get updates on the progress of video saving after you have called finishRecording(). Check the ScreenRecorderEvent **progressPercent** property for the percent value.

message property
message : String
Contains information about the event.

progressPercent property
progressPercent : Number
Contains the percentage of work done when saving the final video as a number from 0 to 100.

VideoSettings class

getStandardVideoSize() static method

public static function getStandardVideoSize(_preferredWidthPix : Number,
                                                  _preferredHeightPix : Number,
                                                  _videoCodec : Number) : Point

Chooses an optimal video frame size for a given codec. The result size is the closest possible to a preferred frame size you specify.

Parameters

_preferredWidthPix - your preferred frame width in pixels
_preferredHeightPix - your preferred frame height in pixels
_videoCodec - the video codec you will use, possible values: VideoSettings.VideoCodecH264 or VideoSettings.VideoCodecJpeg

Returns

A **flash.geom.Point** object, where flash.geom.Point.x represents the video frame **width** and flash.geom.Point.y represents the video frame **height.**

VideoSettings() constructor

public function VideoSettings(
```
    _frameW                   : int = DEFAULT_FRAME_W,
    _frameH                   : int = DEFAULT_FRAME_H,
    _framesPerSecond          : int = DEFAULT_FRAMES_PER_SECOND,
    _videoAverageBitRate      : int = DEFAULT_VIDEO_AVG_BIT_RATE,
    _audioSampleRate          : Number = DEFAULT_AUDIO_SAMPLE_RATE,
    _audioBitRate             : int = DEFAULT_AUDIO_BIT_RATE,
    _audioChannels            : int = DEFAULT_AUDIO_CHANNELS,
    _videoCodec               : Number = DEFAULT_VIDEO_CODEC,
    _jpegQuality              : Number = DEFAULT_JPEG_QUALITY,
    _h264Quality              : Number = DEFAULT_H264_QUALITY,
    _videoCompositionQuality  : Number = DEFAULT_VIDEO_COMPOSITION_QUALITY,
    _inputAudioFilePath       : String = null,
    _audioOffsetSeconds       : int = 0,
    _loopAudio                : Boolean = true,
    _cutOffAudioAtEndOfVideo  : Boolean = true )
```

Creates an instance of VideoSettings.

Parameters

_frameW - width of the recorded video in pixels; default value: DEFAULT_FRAME_W = 640 pixels

_frameH - width of the recorded video in pixels; default value: DEFAULT_FRAME_H = 480 pixels

_framesPerSecond - frame rate of the recorded video; DEFAULT_FRAMES_PER_SECOND = 10

_videoAverageBitRate - average bit rate of the recorded video; default value: DEFAULT_VIDEO_AVG_BIT_RATE = 960000

_audioSampleRate - sample rate for the audio in the final video, in Herz; default value: 12000.0 Hz; has effect only when _inputAudioFilePath is set to a valid file

_audioBitRate - bit rate per audio channel; default value = DEFAULT_AUDIO_BIT_RATE = 64000; has effect only when _inputAudioFilePath is set to a valid file

_audioChannels - number of audio channels in the final video; default value = DEFAULT_AUDIO_CHANNELS = VideoSettings.AudioChannelsStereo; has effect only when _inputAudioFilePath is set to a valid file

_videoCodec - encoder to be used for the recorded video; choose between VideoSettings.VideoCodec264 and VideoSettings.VideoCodecJpeg; default value = DEFAULT_VIDEO_CODEC = VideoSettings.VideoCodec264

_jpegQuality - quality of the JPEG encoding, where 0 = lowest quality 1.0 = highest quality; default value: DEFAULT_JPEG_QUALITY = 0.3; has effect only when videoCodec is set to VideoSettings.VideoCodecJpeg;

_h264Quality - quality of the H.264 encoding; see the h264Quality property for a list of values; default value: DEFAULT_H264_QUALITY = H264Baseline30;has effect only when videoCodec is set to VideoSettings.VideoCodec264;

_videoCompositionQuality - quality preset to be used if the video is mixed with sound after it's been recorded; default value: DEFAULT_VIDEO_COMPOSITION_QUALITY = VideoCompositionPresetMediumQuality; has effect only when _inputAudioFilePath is set to a valid file

_inputAudioFilePath - path to a local audio file (.mp3, .wav, .m4a) to be mixed in the final video; default value: null = no audio

_audioOffsetSeconds - offset in seconds from the start of the video file where audio shoud begin; default value: 0 seconds; has effect only when _inputAudioFilePath is set to a valid file

_loopAudio - a flag, determining whether the audio file should be looped if it's shorter than the final video; defaul value: true; has effect only when _inputAudioFilePath is set to a valid file

_cutOffAudioAtEndOfVideo - a flag, determining whether an audio file, which is longer than the final video, should be cut off at the end of the video; defaul value: true; has effect only when _inputAudioFilePath is set to a valid file

frameWidth property
frameWidth : int
Width of the recorded video in pixels. See getStandardVideoSize() for choosing an optimal video size.

frameHeight property
frameHeight : int
Height of the recorded video in pixels. See getStandardVideoSize() for choosing an optimal video size.

videoAverageBitRate property
videoAverageBitRate : int
Average bit rate of the recorded video.

audioSampleRate property
audioSampleRate : Number
A sample rate, in hertz, expressed as a floating point value.
Has effect only when **inputAudioFilePath** is set to a valid file.

audioBitRate property
audioBitRate : int
Bit rate per audio channel.
Has effect only when **inputAudioFilePath** is set to a valid file.

videoCodec property

`videoCodec : Number`

Encoder to be used for the recorded video.

Possible values:

- **VideoSettings.VideoCodec264**
- **VideoSettings.VideoCodecJpeg**.

jpegQuality property

`jpegQuality : Number`

VideoSettings.VideoCodecJpeg only. Specifies JPEG coded quality.

Use values between **0.0** and **1.0**, where 0.0 = lowest quality, 1.0 = highest quality.

h264Quality property

`h264Quality : Number`

VideoSettings.VideoCodec264 only. Specifies the H.264 encoding quality.

Possible values in increasing level of quality:

- **VideoSettings.H264Baseline30**
- **VideoSettings.H264Baseline31**
- **VideoSettings.H264Baseline41**
- **VideoSettings.H264BaselineAuto** - supported on iOS 7 and newer
- **VideoSettings.H264Main30**
- **VideoSettings.H264Main31**
- **VideoSettings.H264Main32**
- **VideoSettings.H264Main41**
- **VideoSettings.H264MainAuto** - supported on iOS 7 and newer
- **VideoSettings.H264High40** - supported on iOS 6 and newer
- **VideoSettings.H264High41** - supported on iOS 6 and newer
- **VideoSettings.H264HighAuto** - supported on iOS 7 and newer

audioChannels property

`audioChannels : Number`

Determines the audio channel configuration in the recorded video.

Has effect only when **inputAudioFilePath** is set to a valid file.

Possible values:

- **VideoSettings.AudioChannelsMono**
- **VideoSettings.AudioChannelsStereo.**

framesPerSecond property

`framesPerSecond : Number`

Determines the frame rate of the recorded video in frames per second.

videoCompositionQuality property

`videoCompositionQuality : Number`

Sets the quality preset to be used if the video is mixed with sound after it's been recorded.

Has effect only when **inputAudioFilePath** is set to a valid file.

Possible values:

- **VideoSettings.VideoCompositionPresetLowQuality**
- **VideoSettings.VideoCompositionPresetMediumQuality**
- **VideoSettings.VideoCompositionPresetHighestQuality**

scaleMode property

`scaleMode : Number`

Determines the scaling of the DisplayObject in the video frame when a **scaleFactor** has been set.
Possible values:
- **VideoSettings.VIDEO_SCALE_NONE**
- **VideoSettings.VIDEO_SCALE_FIT_W**
- **VideoSettings.VIDEO_SCALE_FIT_H**
- **VideoSettings.VIDEO_SCALE_FIT_ALL**

scaleFactor property
`scaleFactor : Point`
Determines the scaling of the DisplayObject in the video frame.

inputAudioFilePath property
`inputAudioFilePath : String`
Determines the path to the audio file that should be mixed in the video.
Leave to null, if you don't want to use audio.

audioOffsetSeconds property
`audioOffsetSeconds : int`
Offset in seconds from the start of the video file where audio shoud begin.
Has effect only when **inputAudioFilePath** is set to a valid file.

loopAudio property
`loopAudio : Boolean`
A flag, determining whether an audio file, which is longer than the final video, should be cut off at the end of the video.
Has effect only when **inputAudioFilePath** is set to a valid file.

cutOffAudioAtEndOfVideo property
`cutOffAudioAtEndOfVideo : Boolean`
A flag, determining whether an audio file, which is longer than the final video, should be cut off at the end of the video.
Has effect only when **inputAudioFilePath** is set to a valid file.

Building the ANE

If you have purchased the source code for **DiaDrawScreenCast.ane**, you have the option of modifying the extension to meet your specific needs.

Here is how to build it.

Modify local.properties

Open **DiaDrawScreenCastANE/AIRLibrary/build scripts/local.properties** and set IOS_SDK and FLEX_HOME to point to the SDK installs on your machine:

```
IOS_SDK=/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/
Developer/SDKs/iPhoneOS7.0.sdk

FLEX_HOME=/Applications/Adobe Flash Builder 4.6/sdks/4.6.0

MXMLC=${FLEX_HOME}/bin/mxmlc

ADT=${FLEX_HOME}/bin/adt
```

Option 1: On the command line

On the command line navigate to **DiaDrawScreenCastANE/AIRLibrary/build scripts/** and run

```
ant
```

Option 2: In Flash Biulder

Import **DiaDrawScreenCastANE/ AIRLibrary** in Flash Builder and select **Project** > **Properties**.

Select Builders, then click **New…** to create a new **Ant Builder**.

Inside the **Edit launch configuration properties** dialog click Browse Workspace to navigate to **DiaDrawScreenCastANE/AIRLibrary/ build scripts/** and select **build.xml**, then **OK** the dialog. Back in Flash Builder select **Project** > **Build Project**. This should rebuild everything from scratch and package an ANE file in **DiaDrawScreenCastANE/ane**.
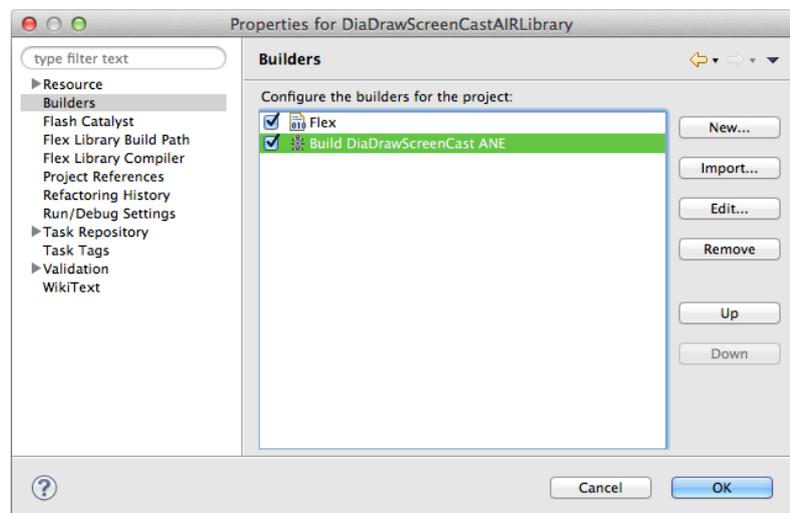
Questions?

Drop us a line at support@diadraw.com if you have any questions about using **DiaDrawSCreenCast.ane** or this manual.

What else is out there?

Find more ANE-related information in our tutorials and articles.

Have a look at the other ANEs on our website.

Have you checked out our Easy Native Extensions eBook yet?
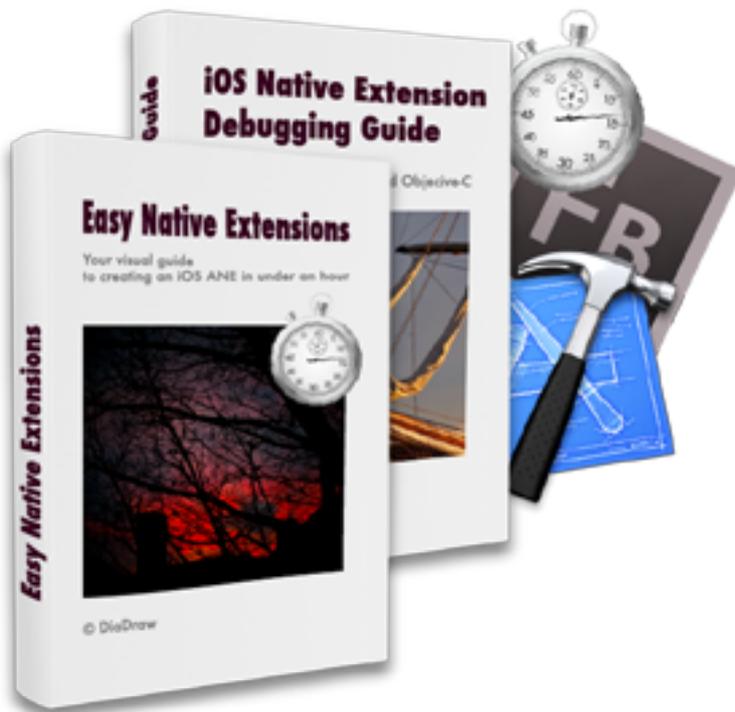
Thank you

… and happy coding from The DiaDraw Team!

Developed in collaboration with: **Jonathan Kaye, PhD, President, CommandSim**

http://www.commandsim.com

http://www.simsushare.com