Xcode Core Concepts

Session 401

Anders Bertelrud
Xcode Engineer

Rick Ballard
Xcode Engineer

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

Unlimited Undo
& Redo
Color palette
Precise control
over font, text
style, & paragraph
settings
Writing
tools
PROJECT: APPLICATION.APP
ARCHITECT: XCODE #1

What We'll Cover

- The user interface

What We'll Cover

- The user interface
  - Navigating the workspace

What We'll Cover

- The user interface
  - Navigating the workspace
  - Working effectively with editors

What We'll Cover

- The user interface
  - Navigating the workspace
  - Working effectively with editors
  - Source editor power tricks

What We'll Cover

- The user interface
  - Navigating the workspace
  - Working effectively with editors
  - Source editor power tricks
  - Documentation integration

What We'll Cover

- The user interface
  - Navigating the workspace
  - Working effectively with editors
  - Source editor power tricks
  - Documentation integration
  - Customizing Xcode's behaviors

What We'll Cover

- Fundamental concepts

What We'll Cover

- Fundamental concepts
  - Workspaces

What We'll Cover

- Fundamental concepts
  - Workspaces
  - Projects

What We'll Cover

- Fundamental concepts
  - Workspaces
  - Projects
  - Targets

What We'll Cover

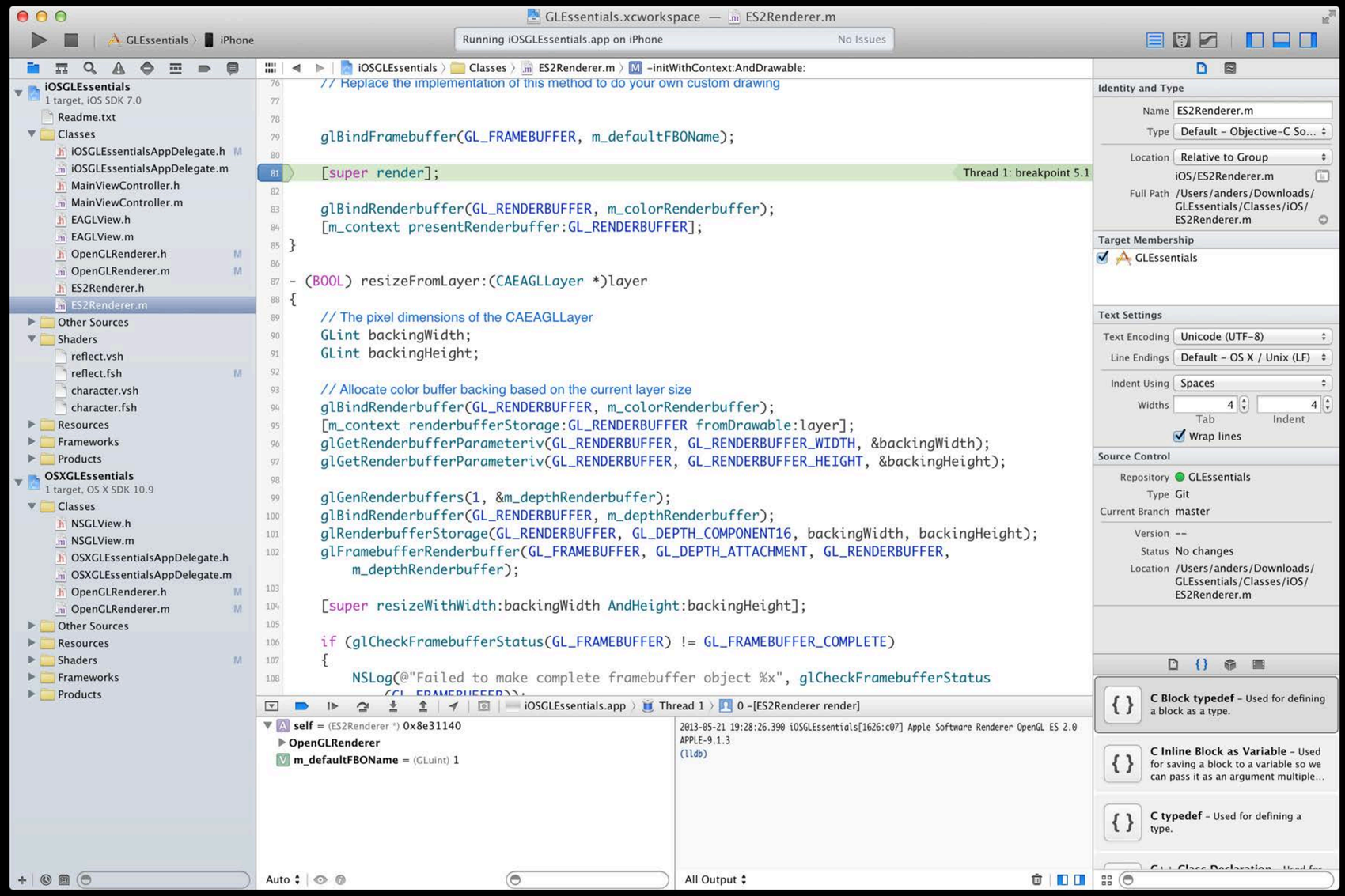
- Fundamental concepts
  - Workspaces
  - Projects
  - Targets
  - Schemes

What We'll Cover

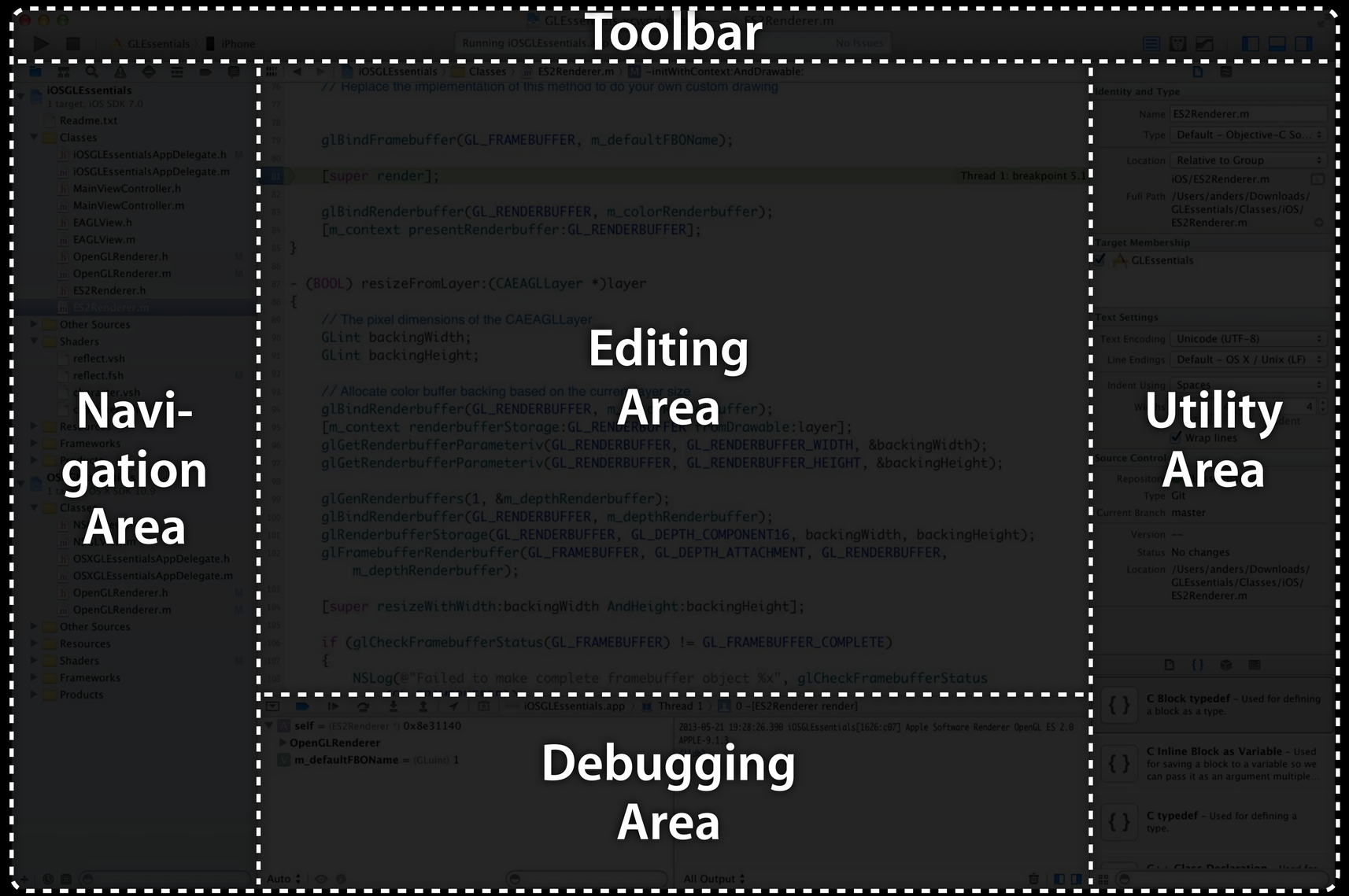
- Fundamental concepts
  - Workspaces
  - Projects
  - Targets
  - Schemes
  - Run destinations

The User Interface

The Workspace Window

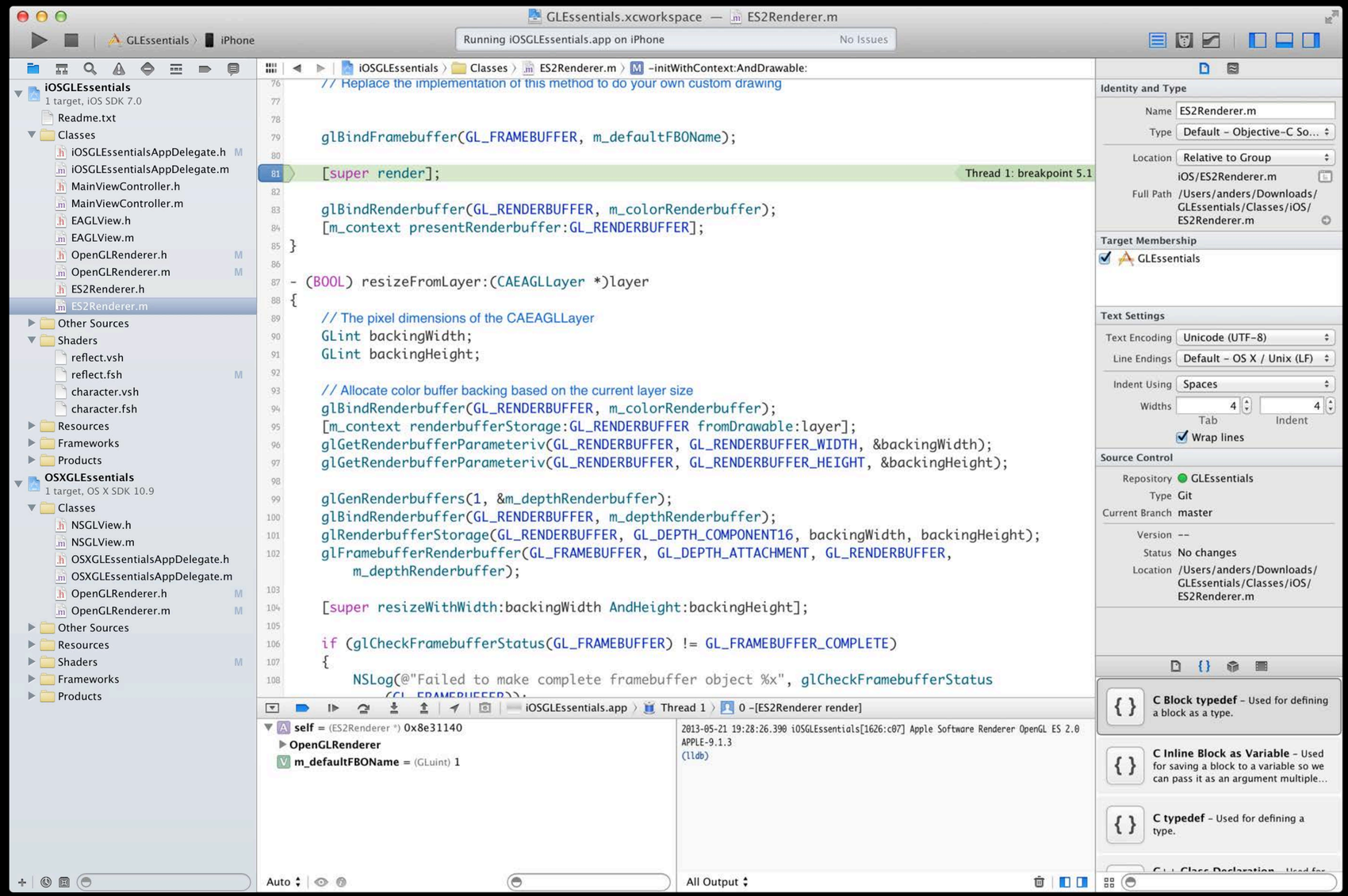


The Workspace Window

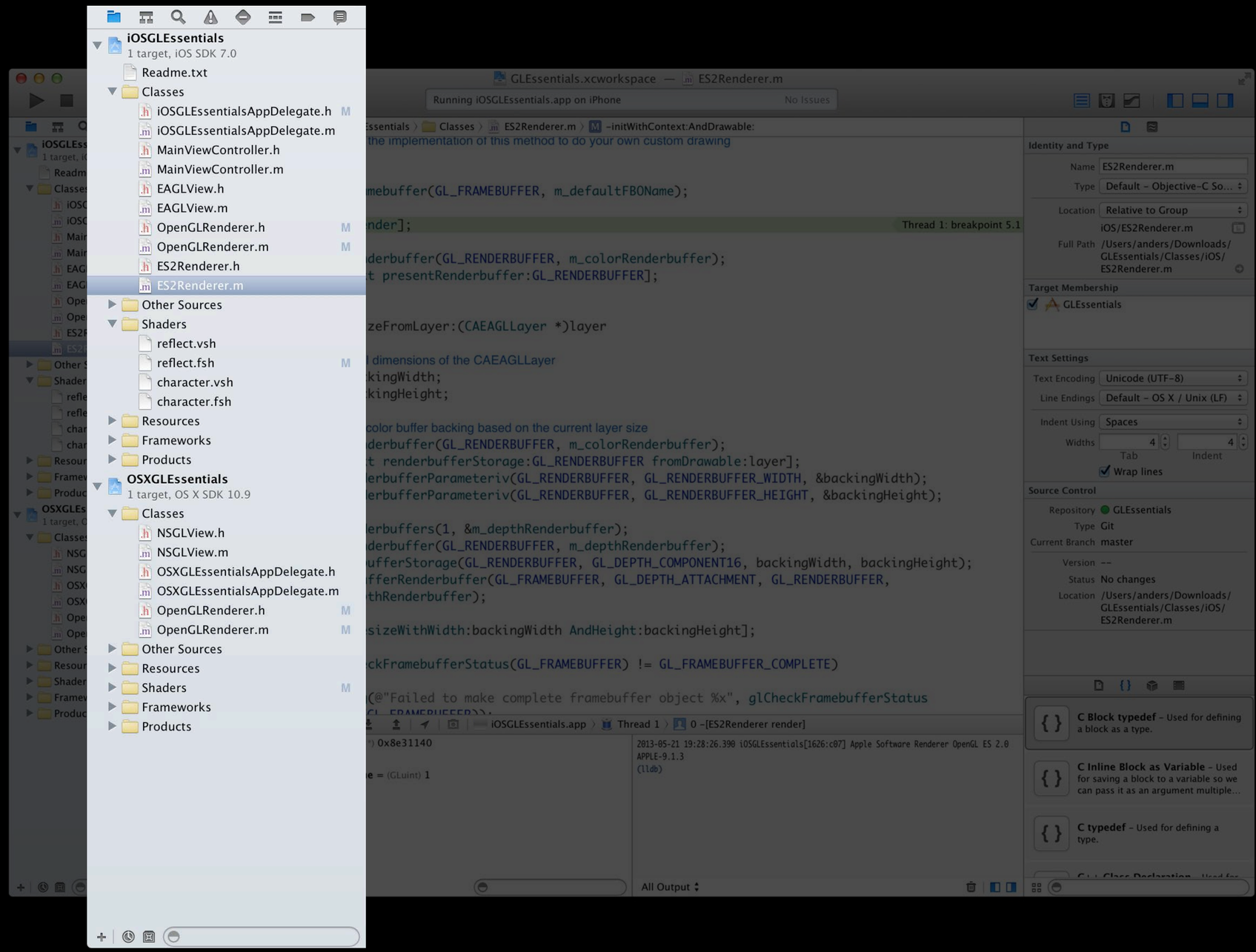


Navigating Your Xcode Workspace

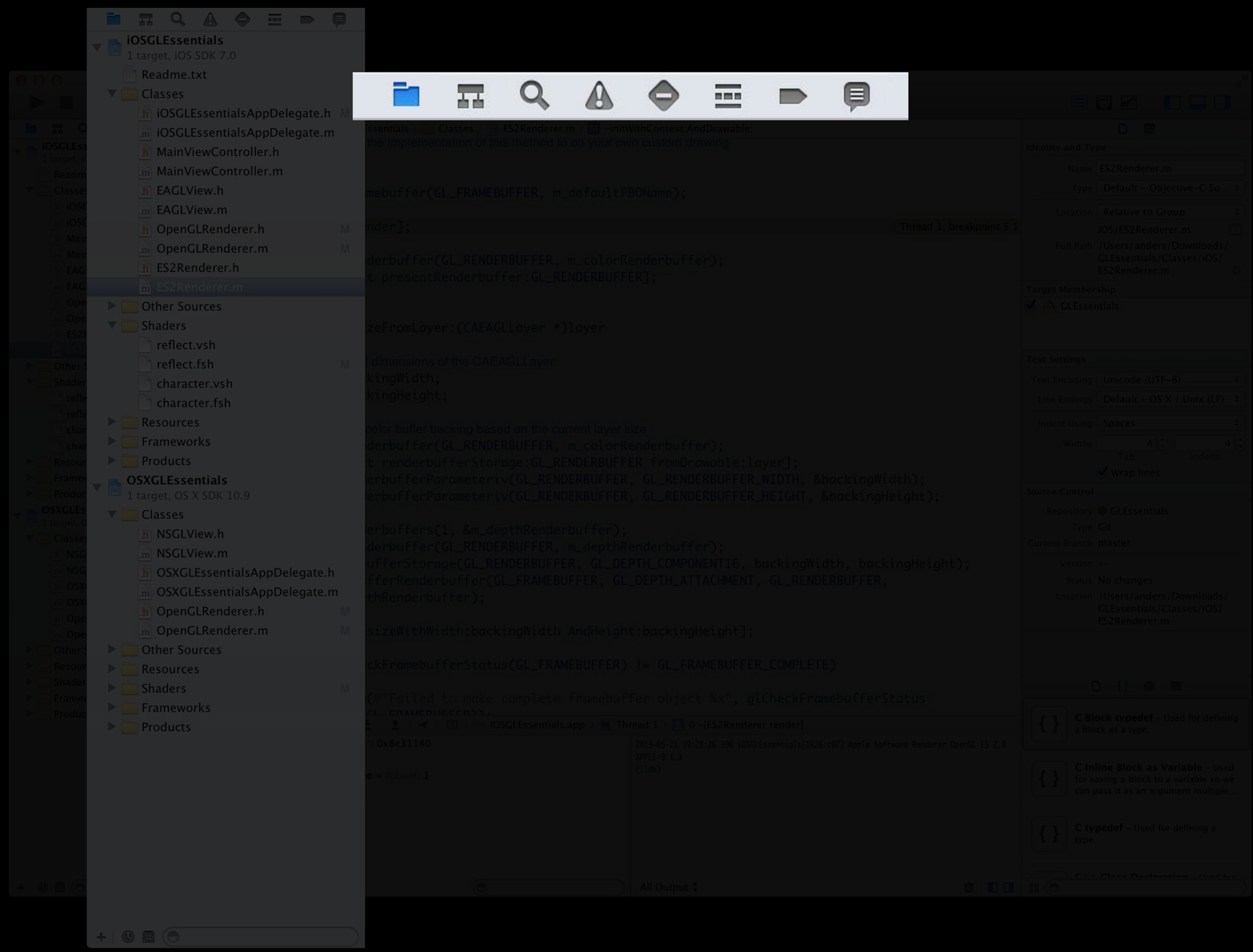
Navigator Area



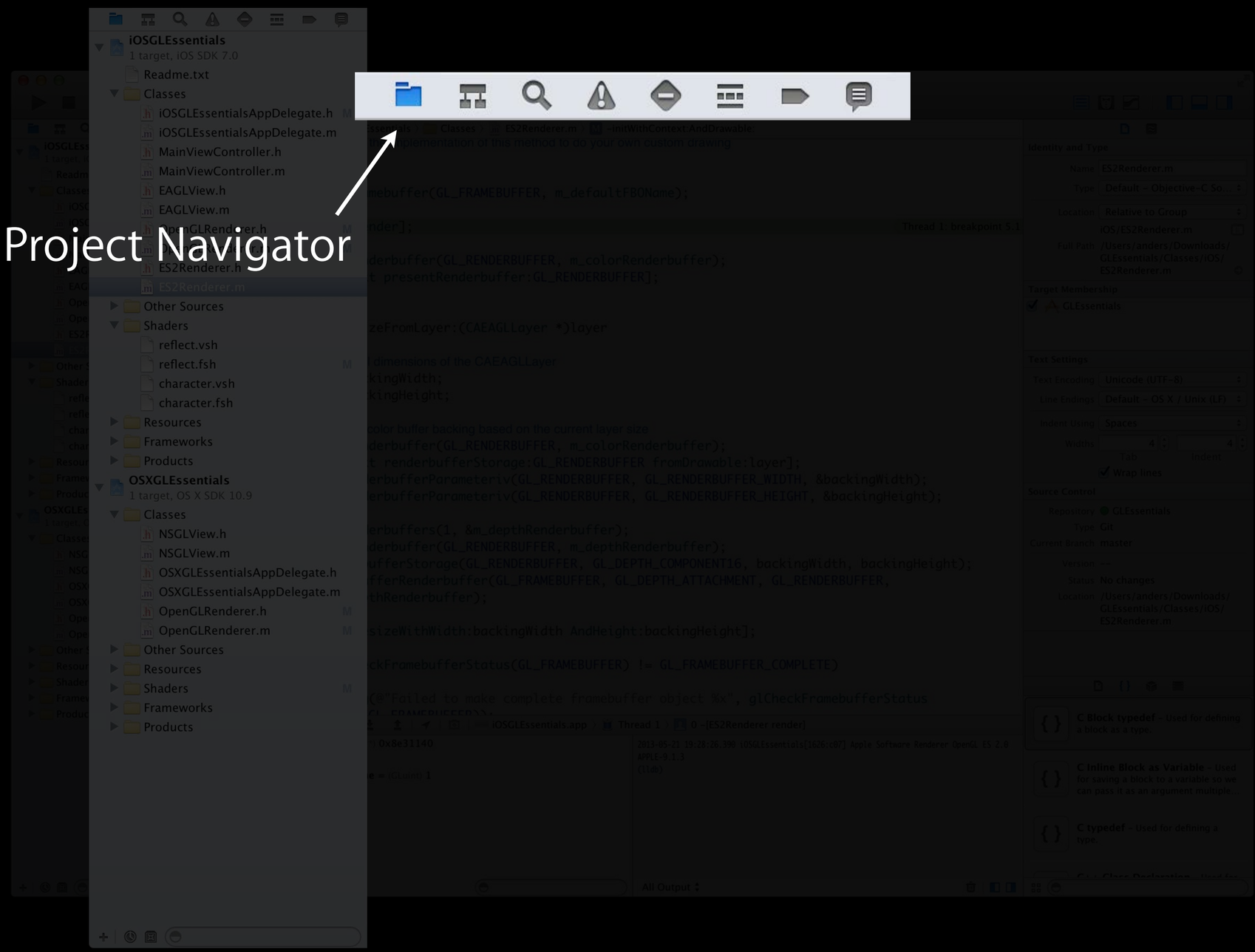
Navigator Area



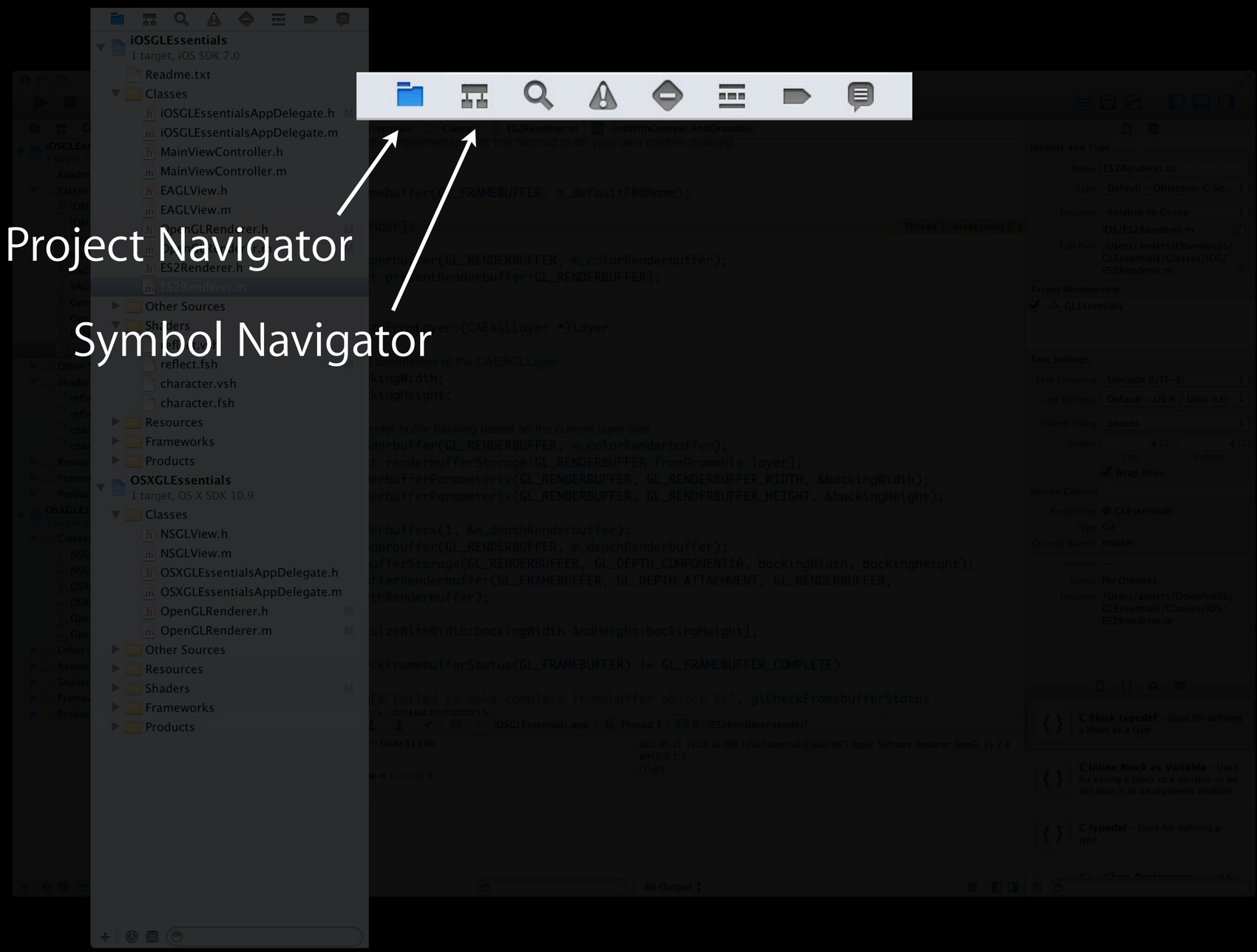
Navigator Area



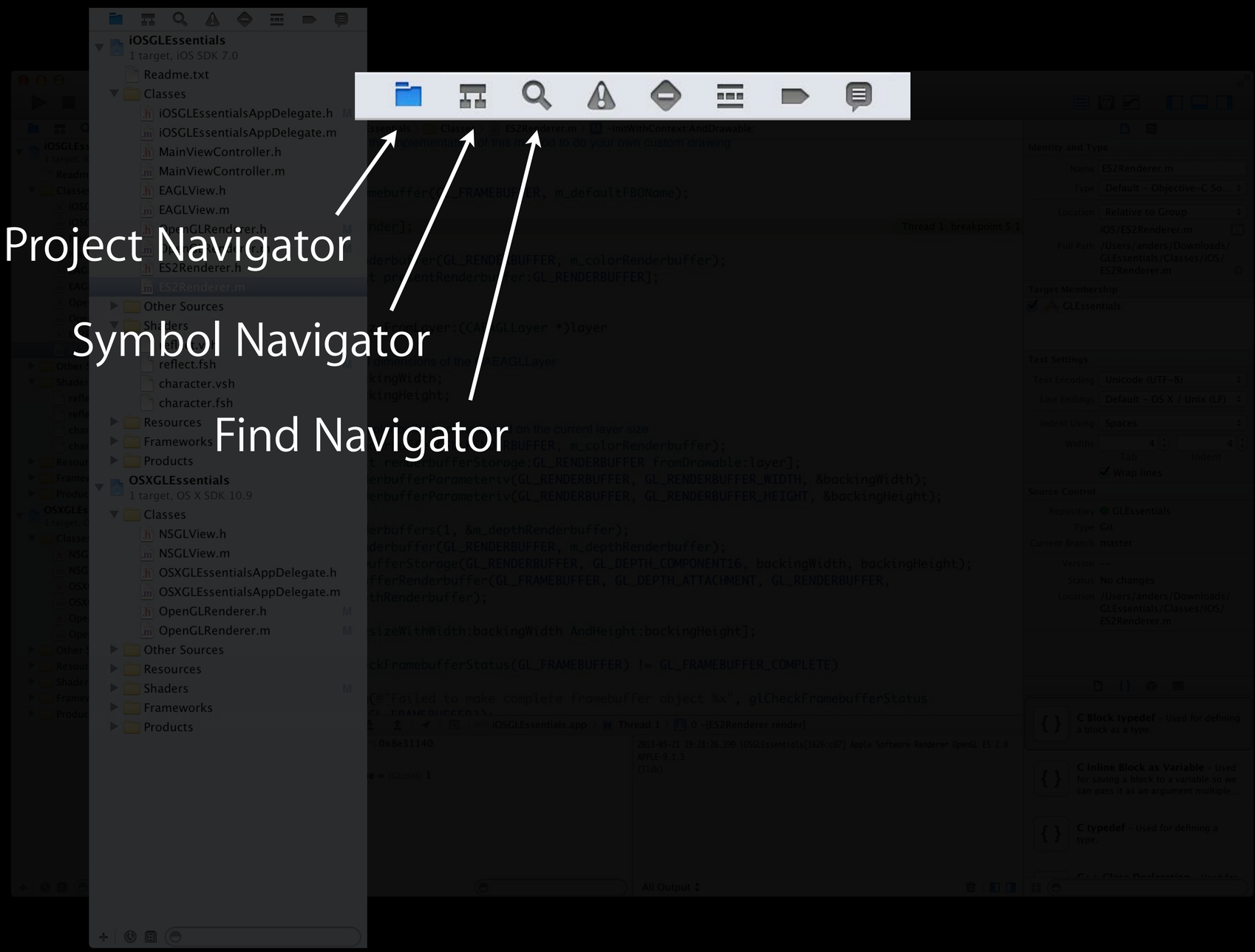
Navigator Area



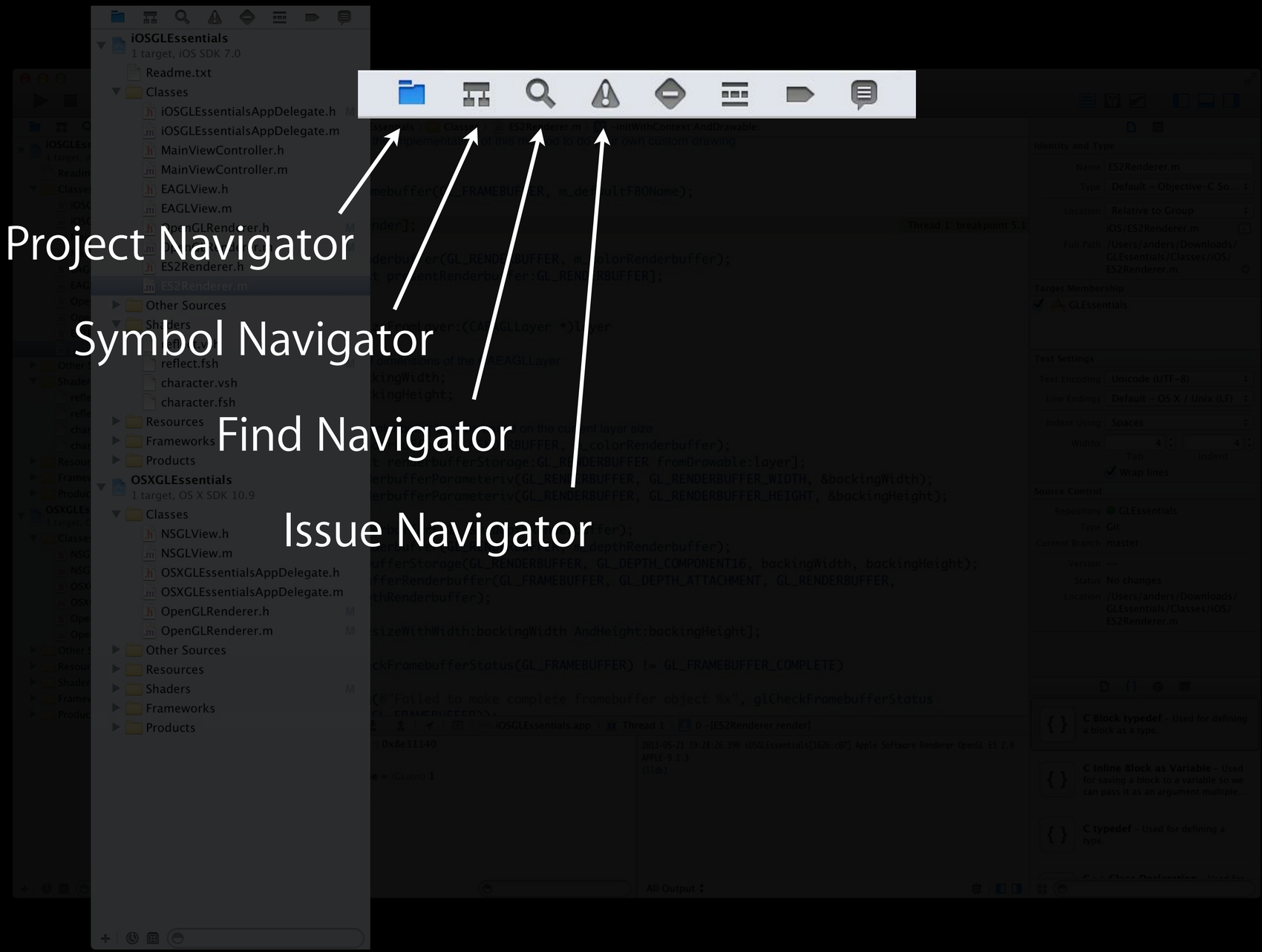
Navigator Area



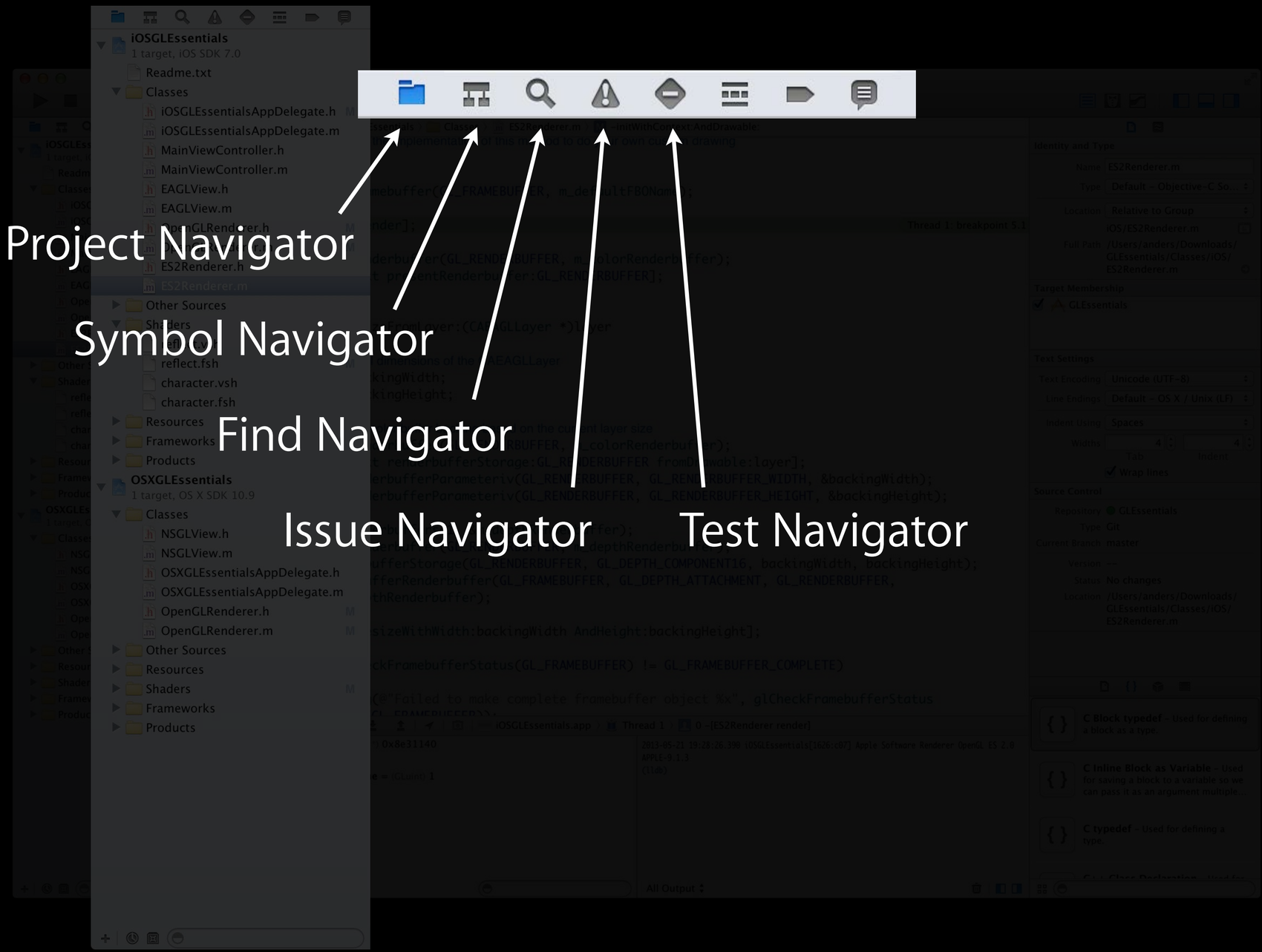
Navigator Area



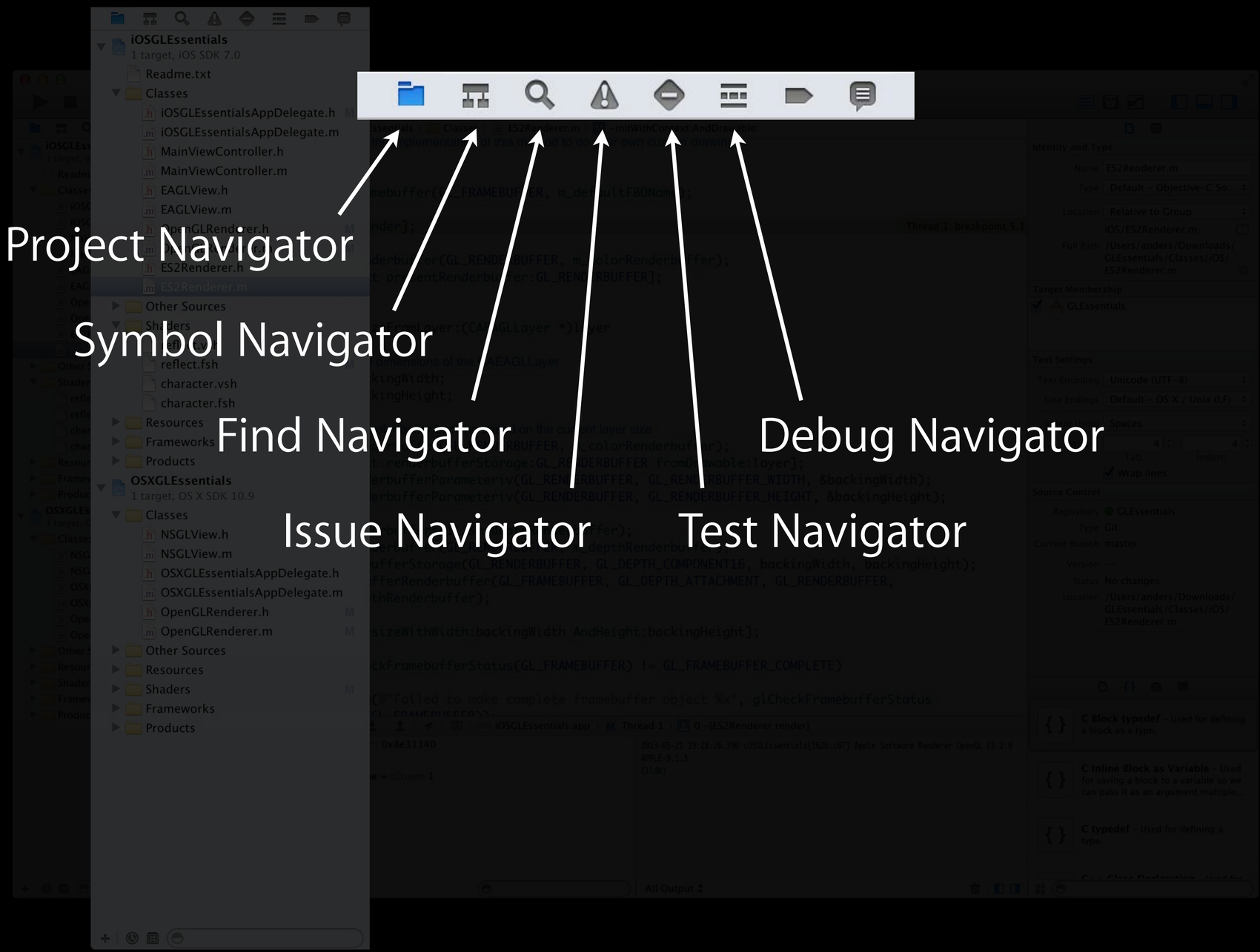
Navigator Area



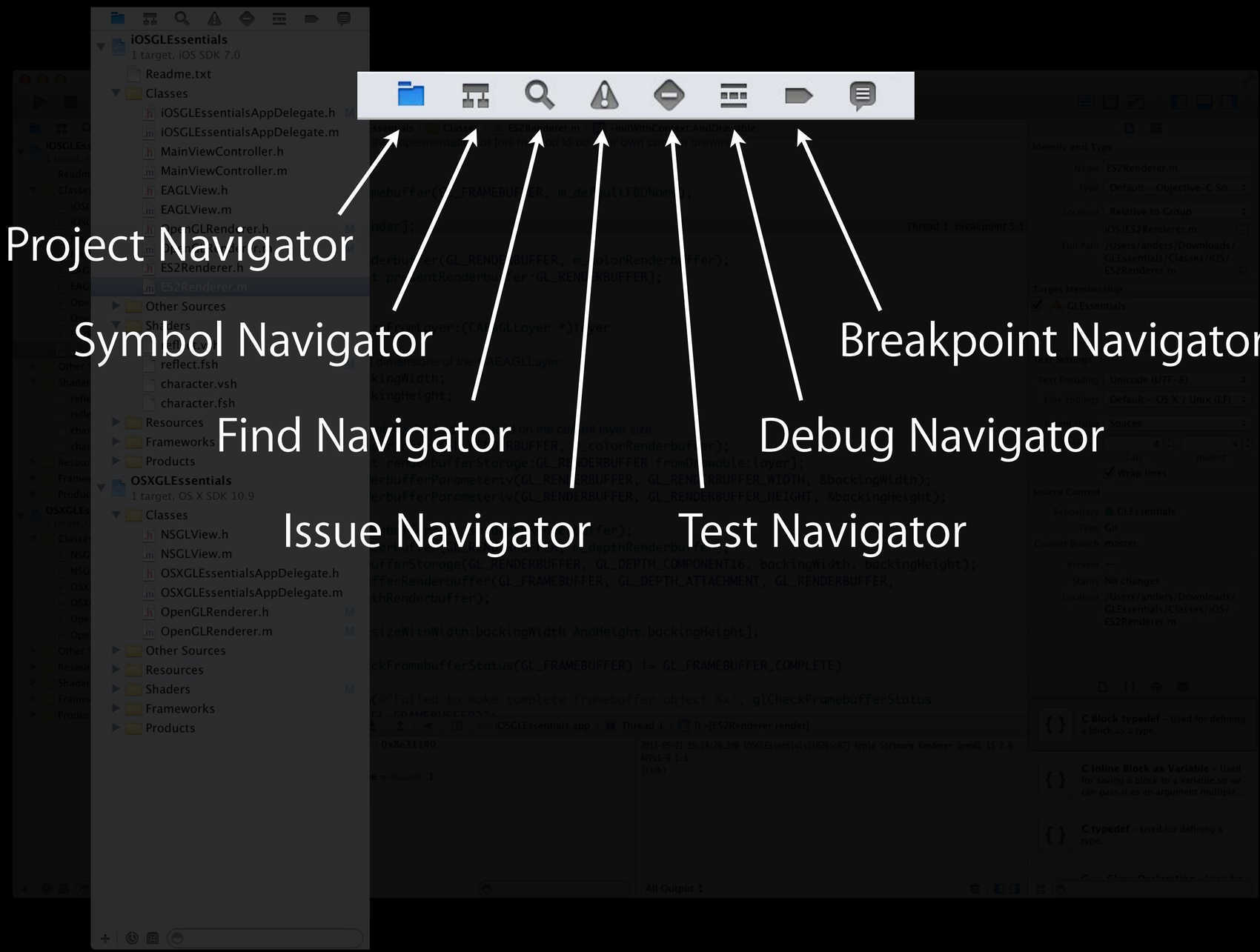
Navigator Area



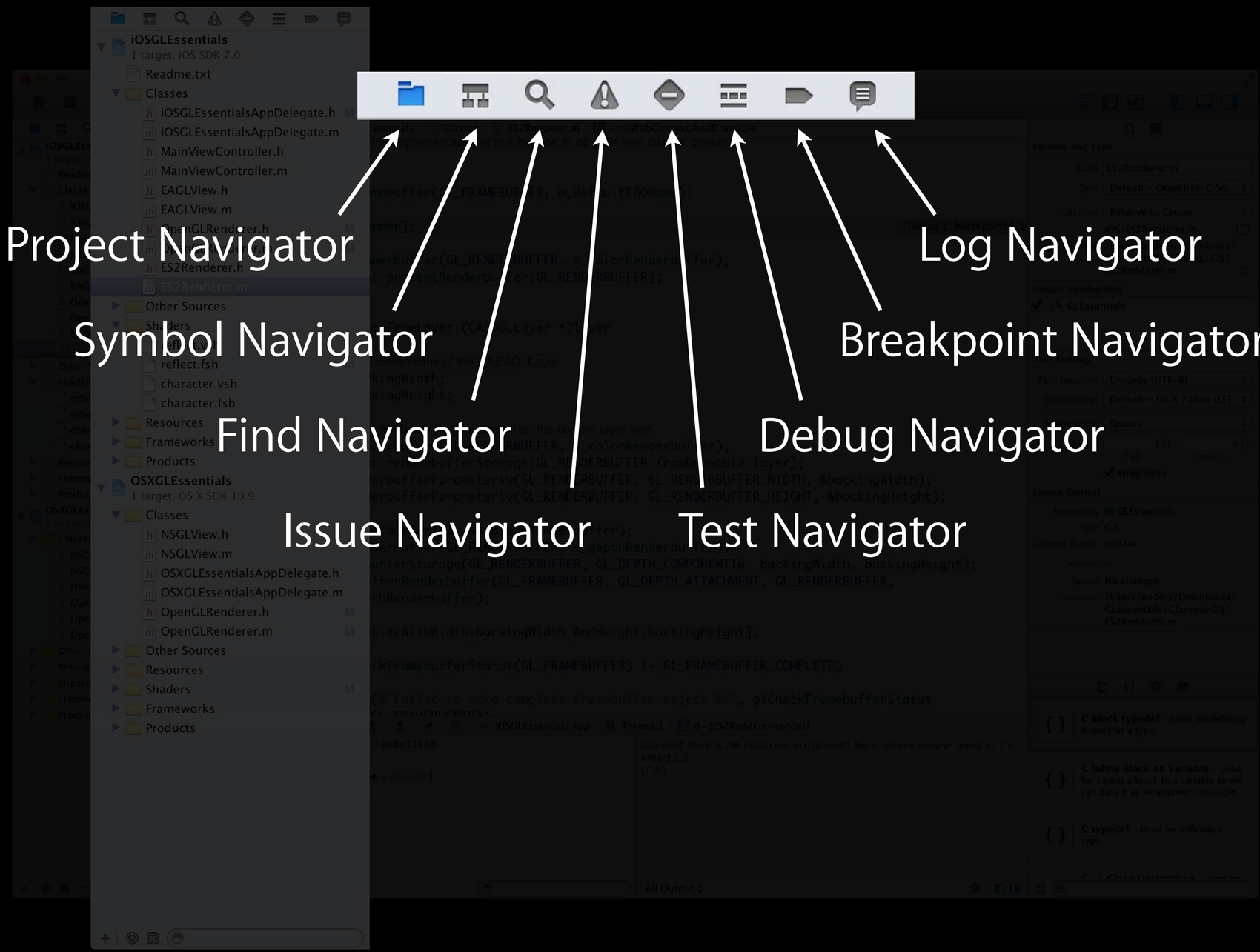
Navigator Area



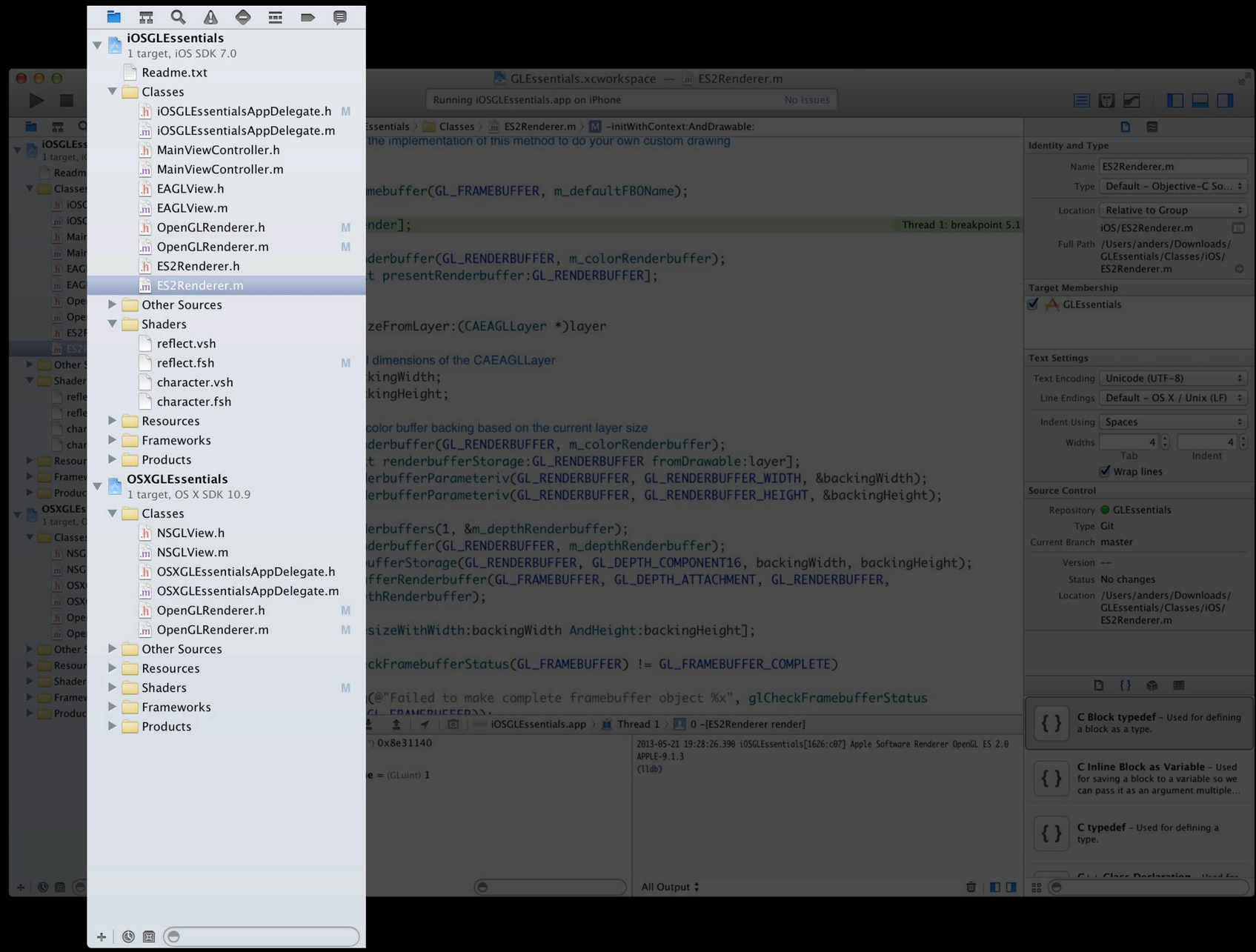
Navigator Area



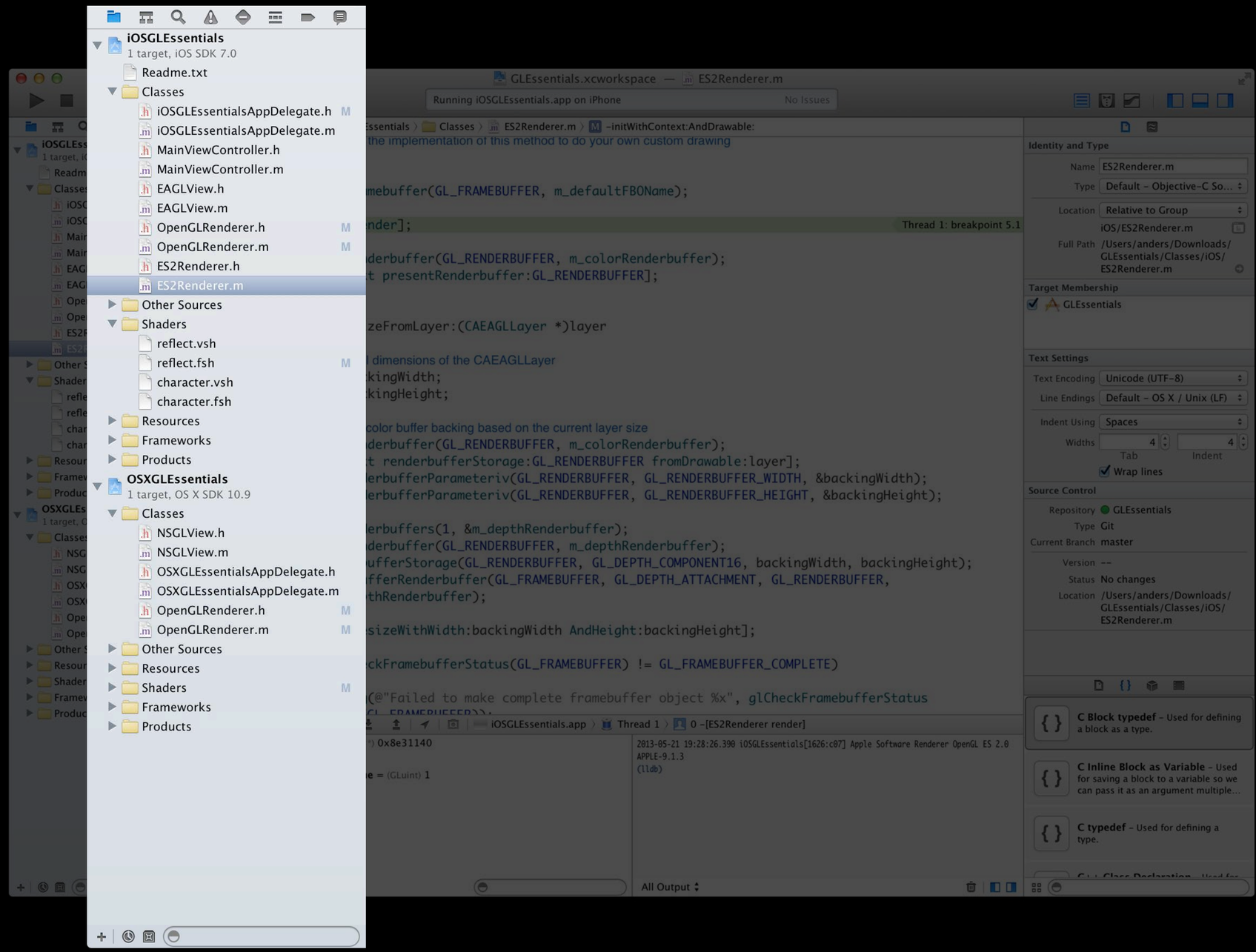
Navigator Area



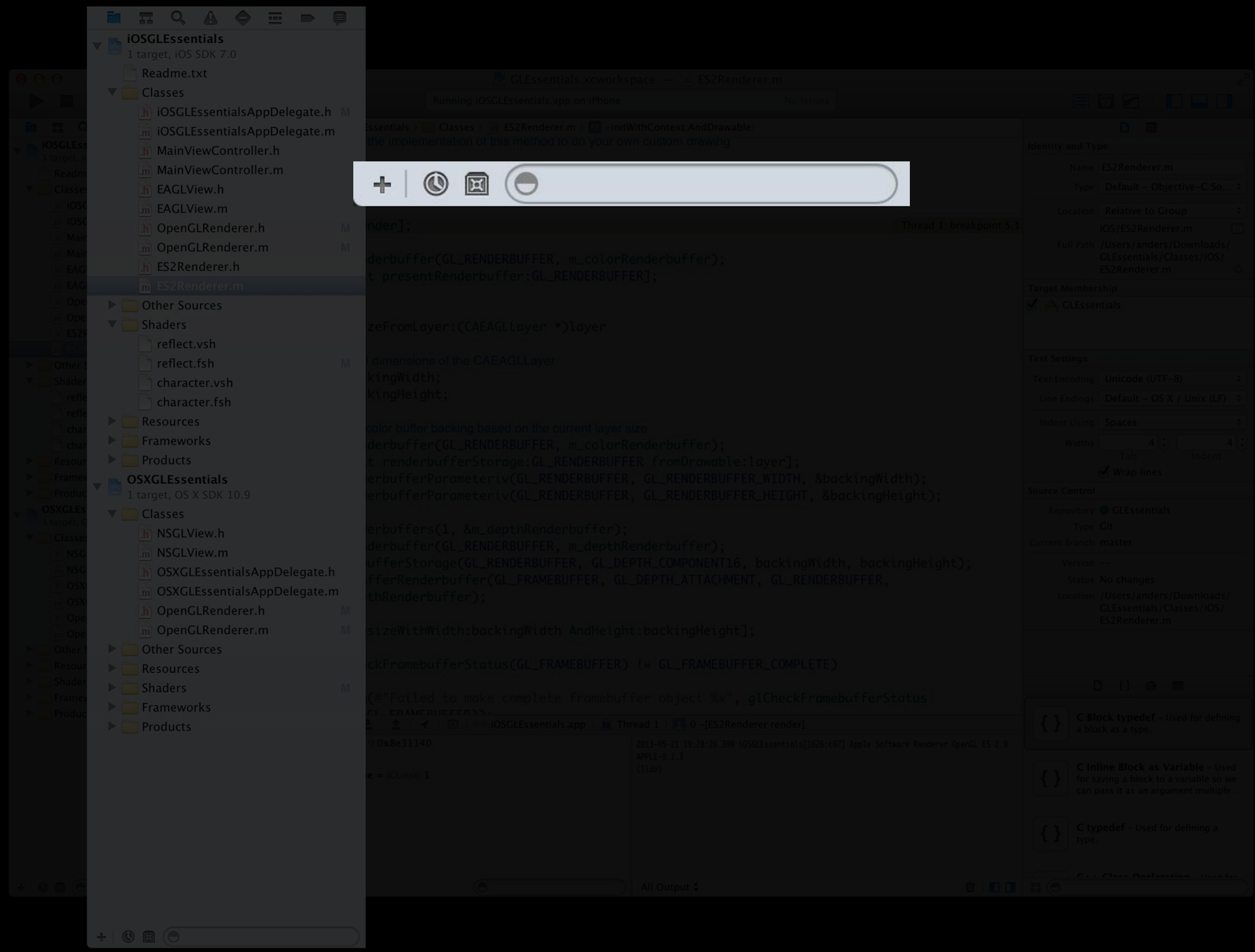
Navigator Area



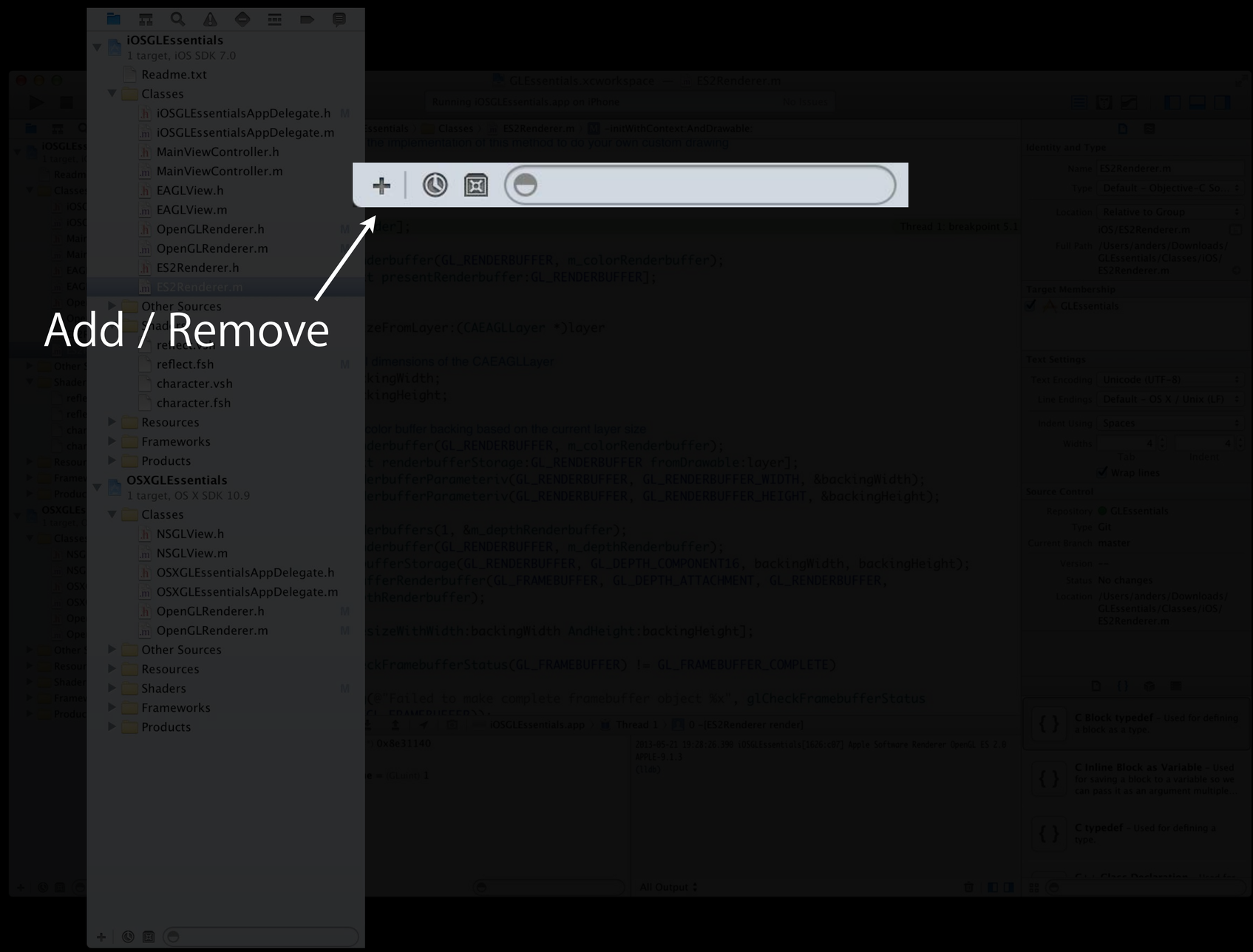
Navigation Filters



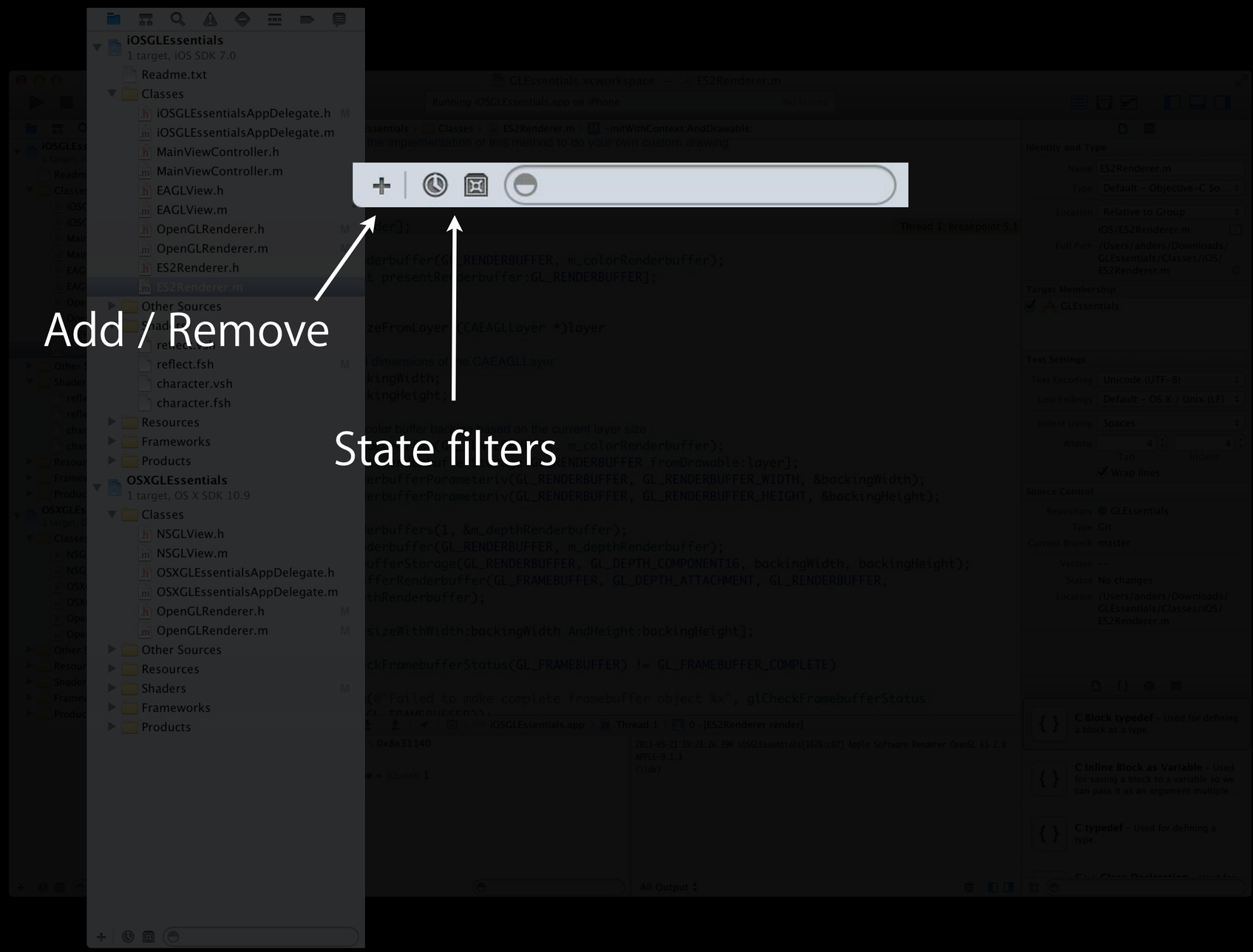
Navigation Filters



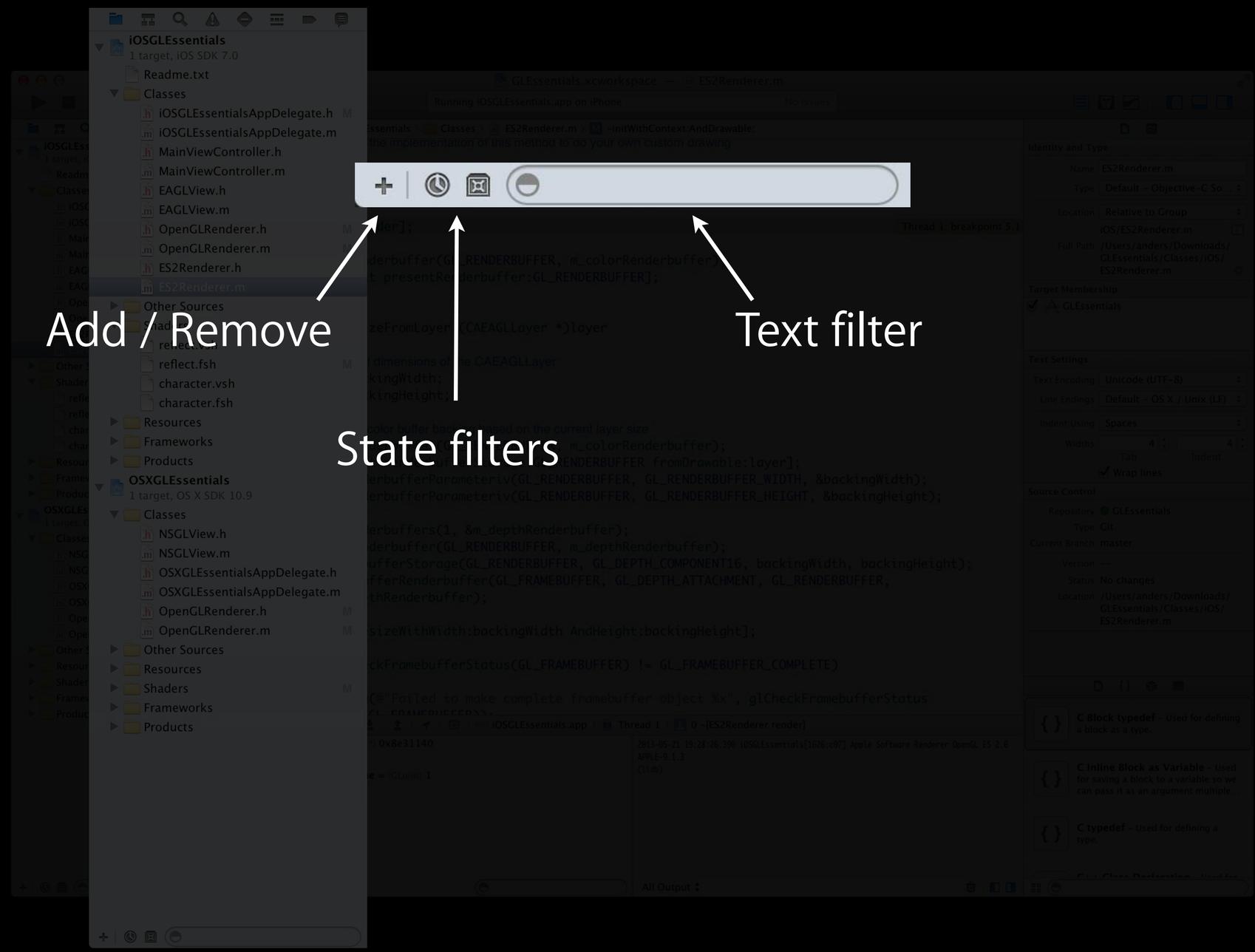
Navigation Filters



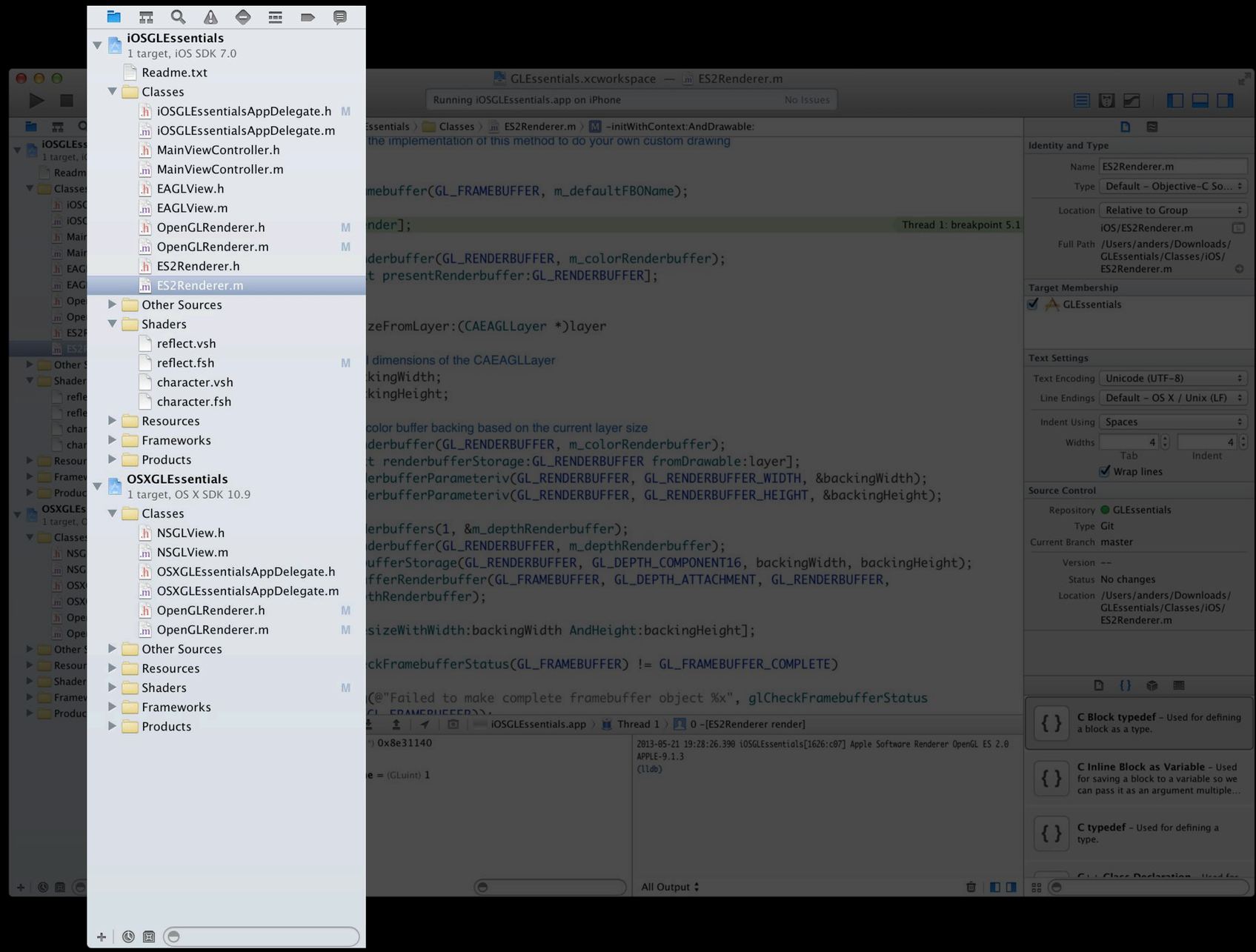
Navigation Filters



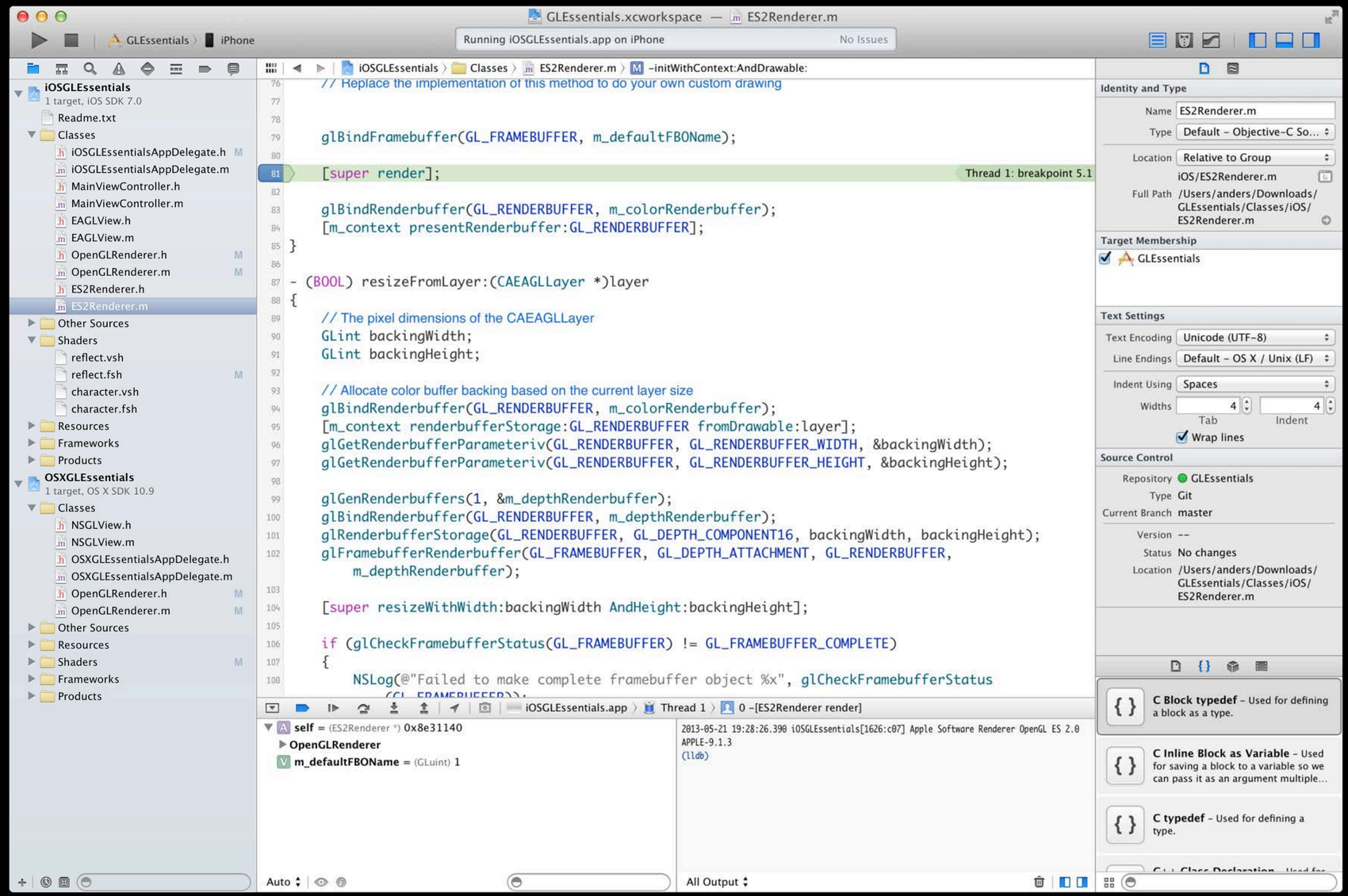
Navigation Filters



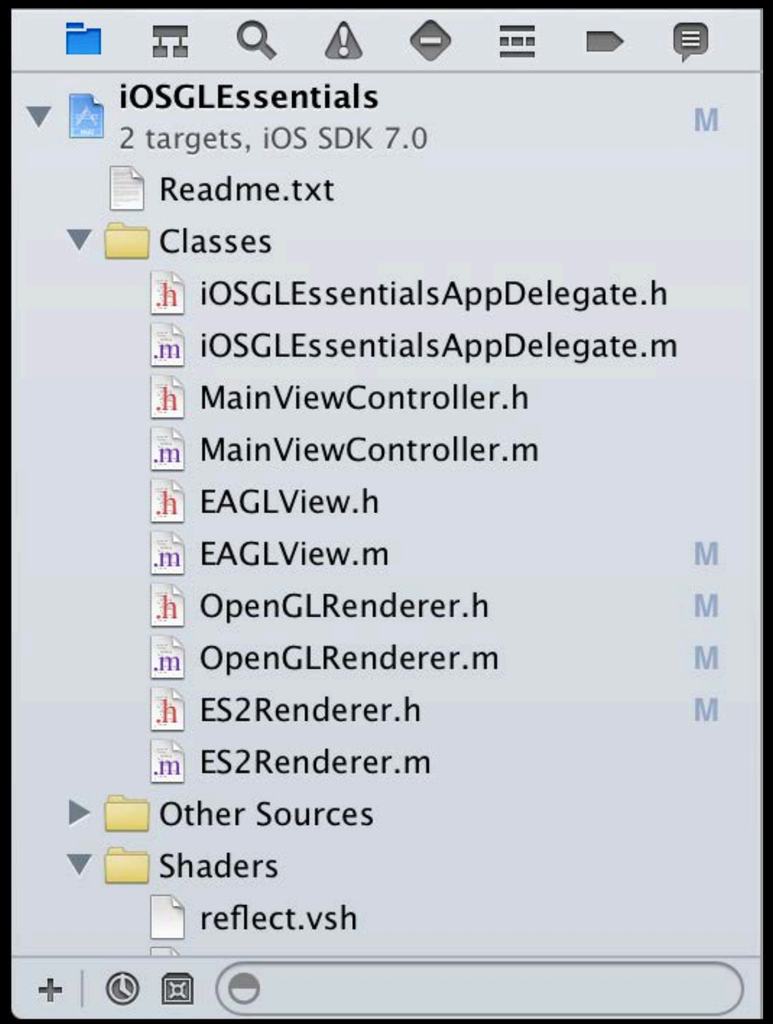
Navigation Filters



Navigation Filters



Project Navigator



Shows a reference hierarchy of projects, files, folders, …

Project Navigator

The “recently edited” filter lets you focus on your working set of files

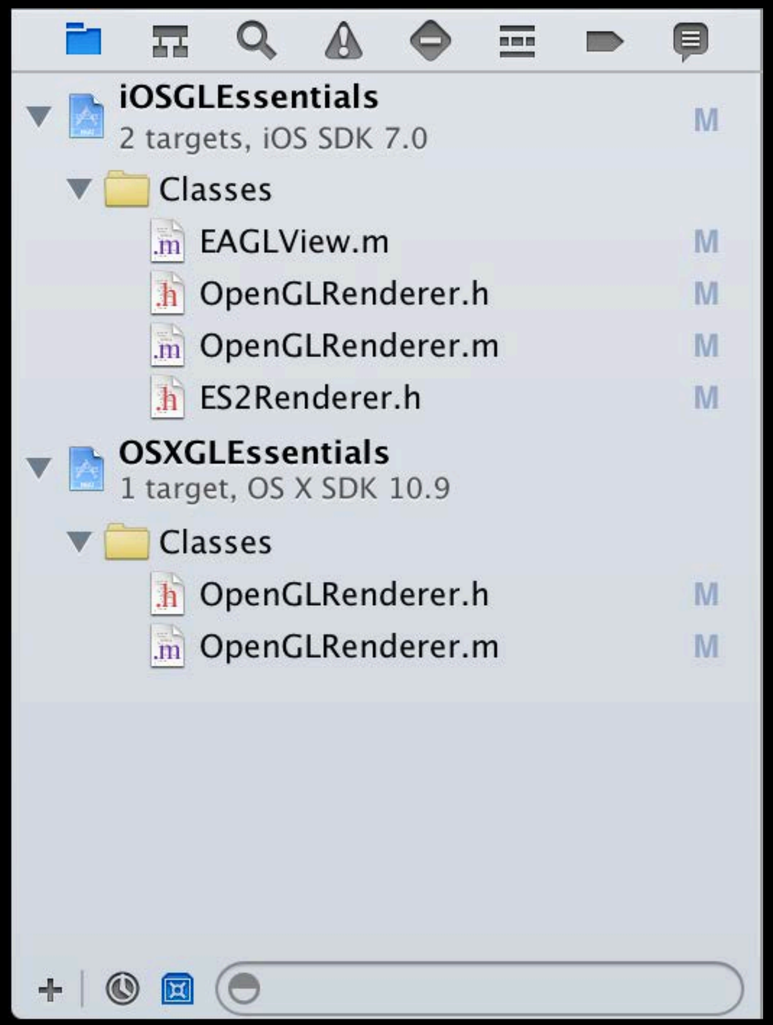
Project Navigator

The “recently edited” filter lets you focus on your working set of files

Project Navigator

The “recently edited” filter lets you focus on your working set of files

Project Navigator

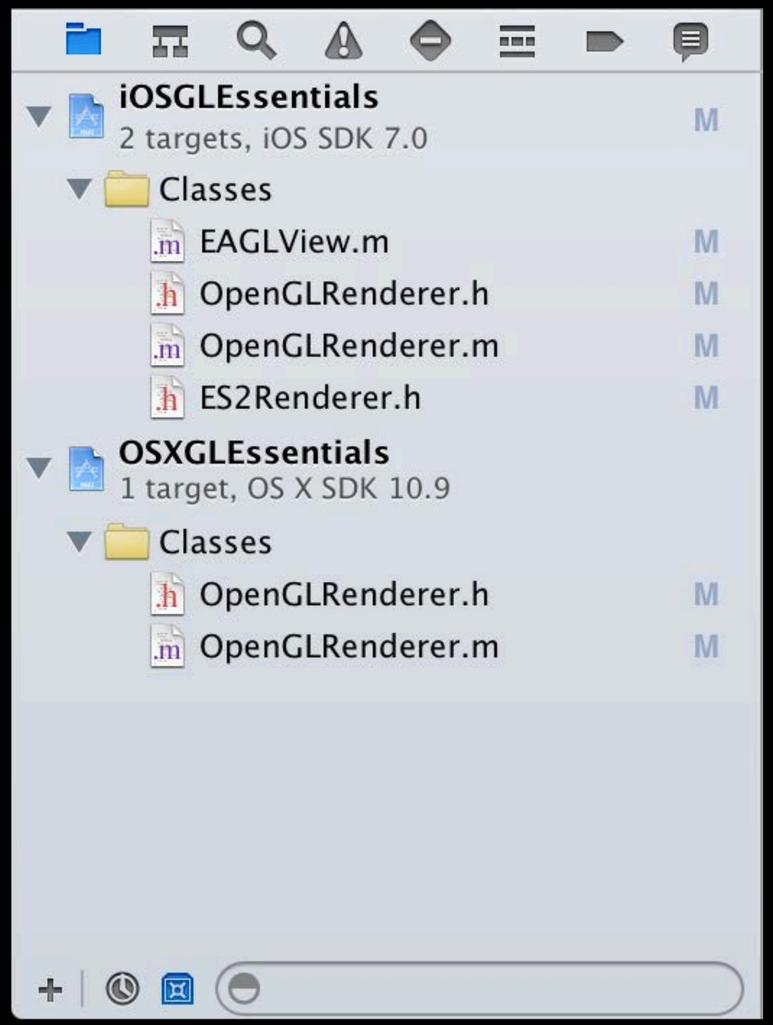


The “modified files” filter lets you focus on your SCM changes

Project Navigator

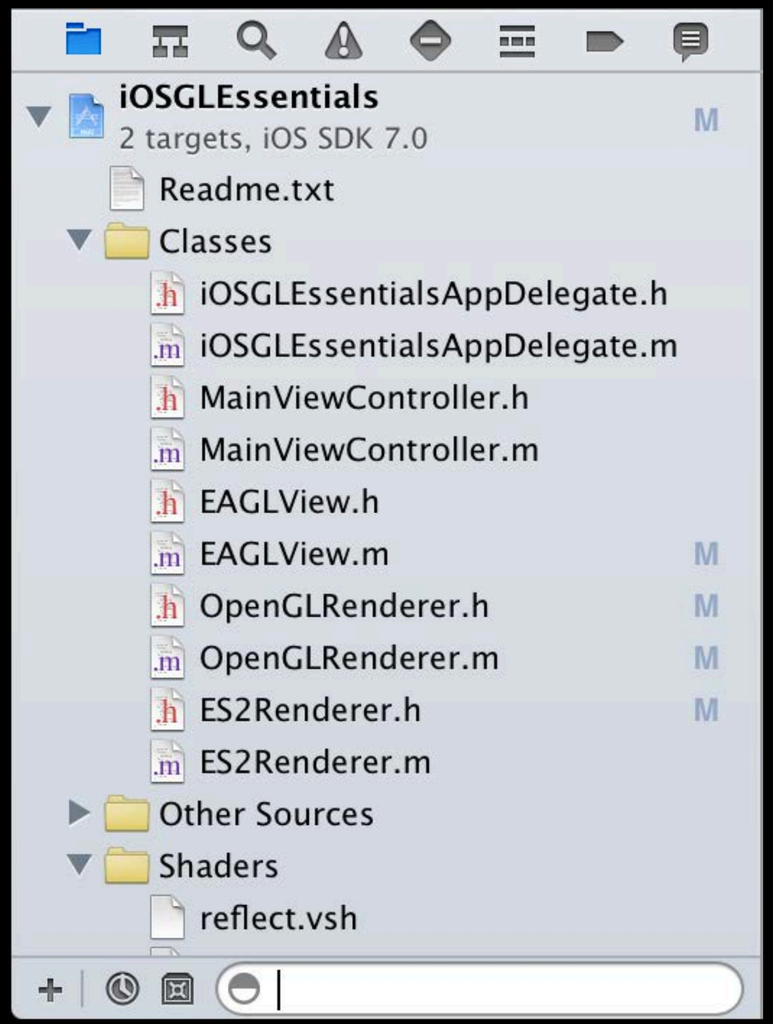
The “modified files” filter lets you focus on your SCM changes

Project Navigator



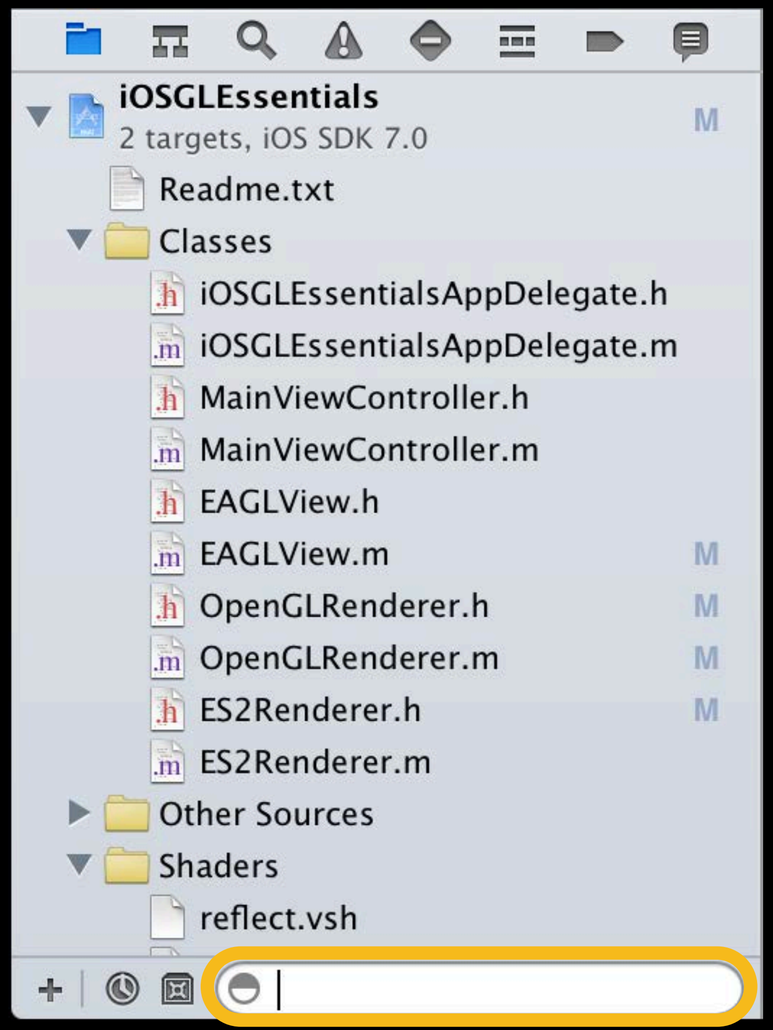
The “modified files” filter lets you focus on your SCM changes

Structure Navigator



Filter on name when you know the name of what you're looking for

Structure Navigator



Filter on name when you know the name of what you're looking for

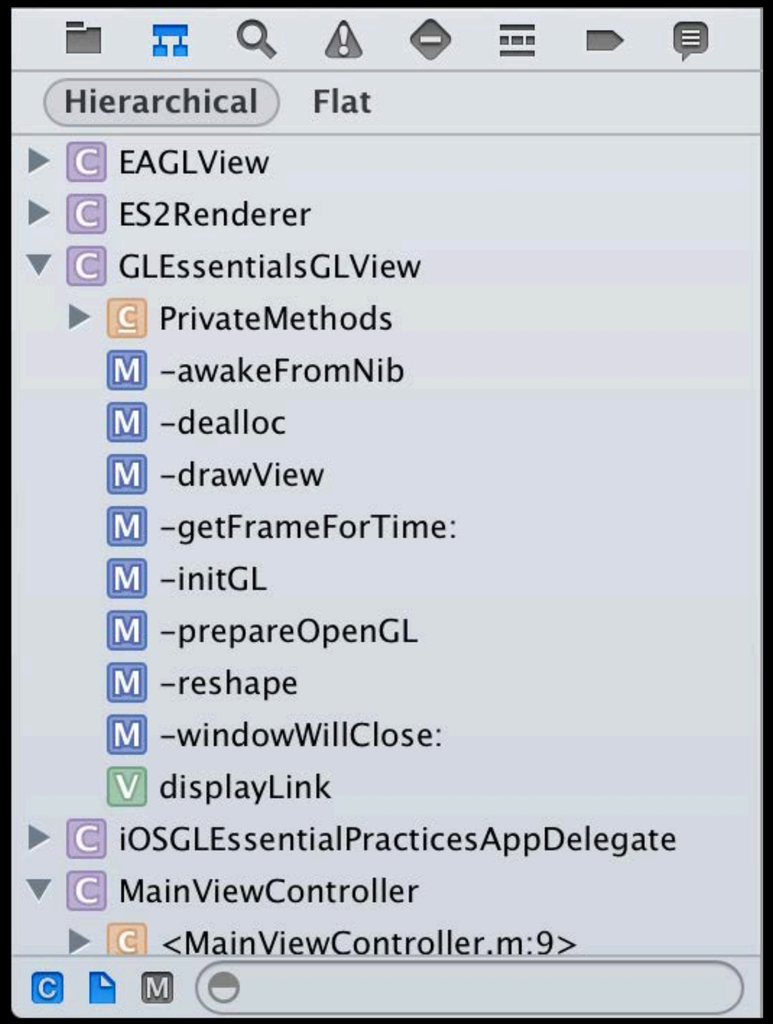
Structure Navigator

Filter on name when you know the name of what you're looking for

Structure Navigator

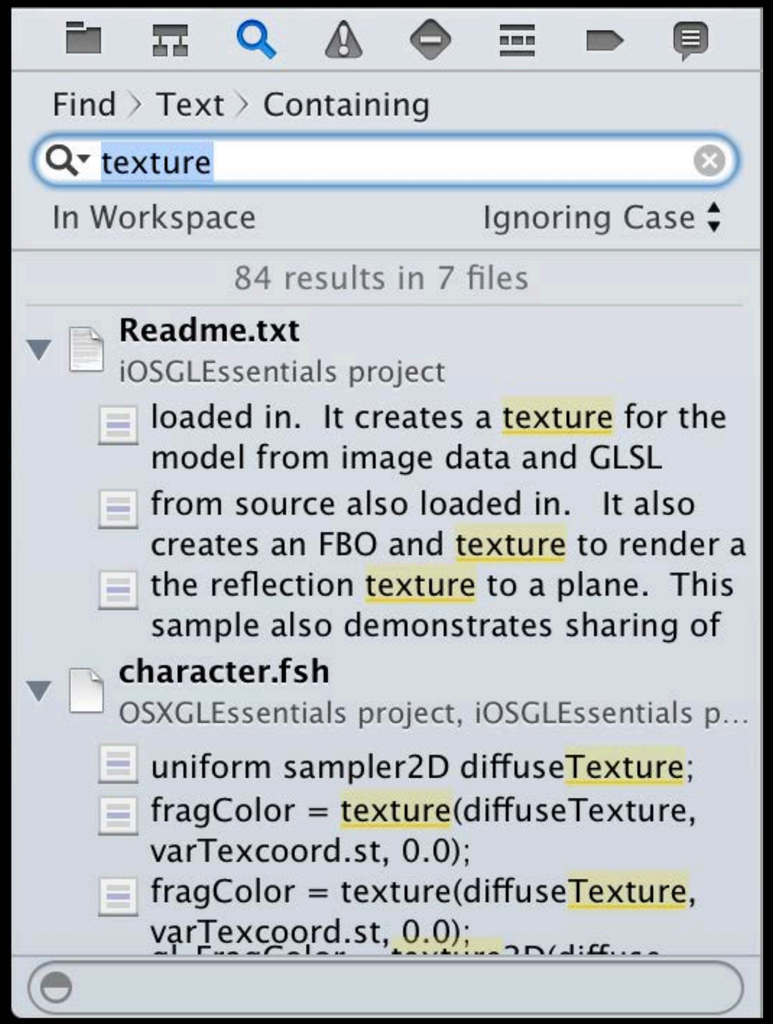
Filter on name when you know the name of what you're looking for

Symbol Navigator



Shows classes, methods, globals, structures, and functions

Find Navigator

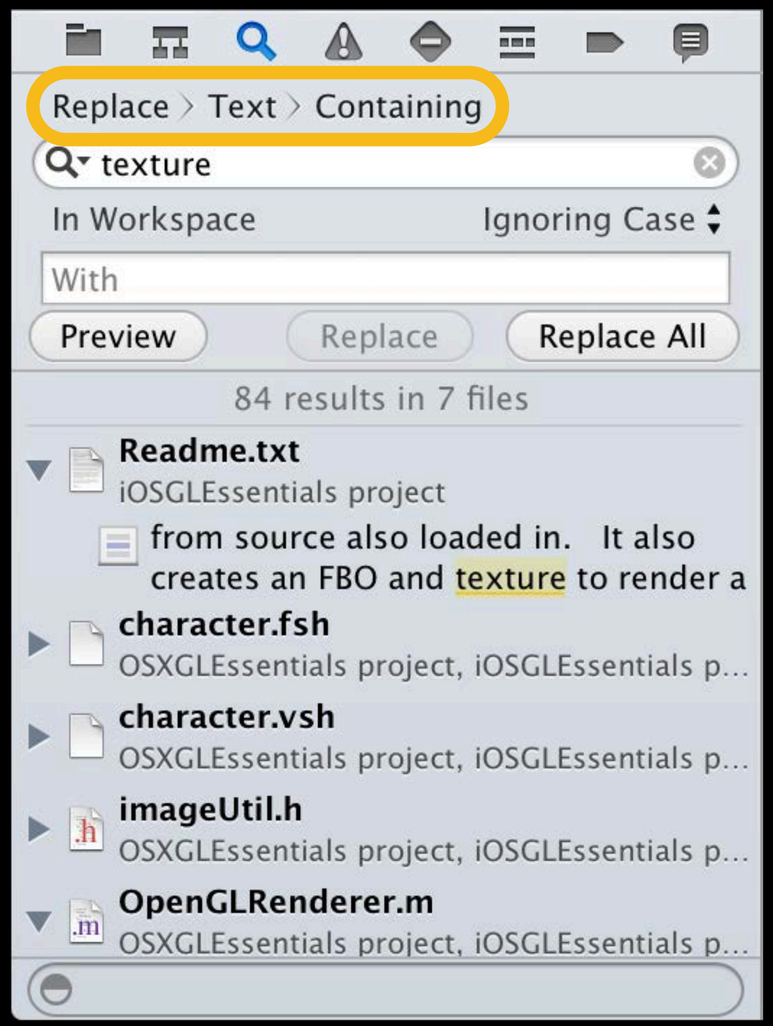


Find or replace plain text, regular expressions, etc.

Find Navigator

Find or replace plain text,
regular expressions, etc.

Find Navigator

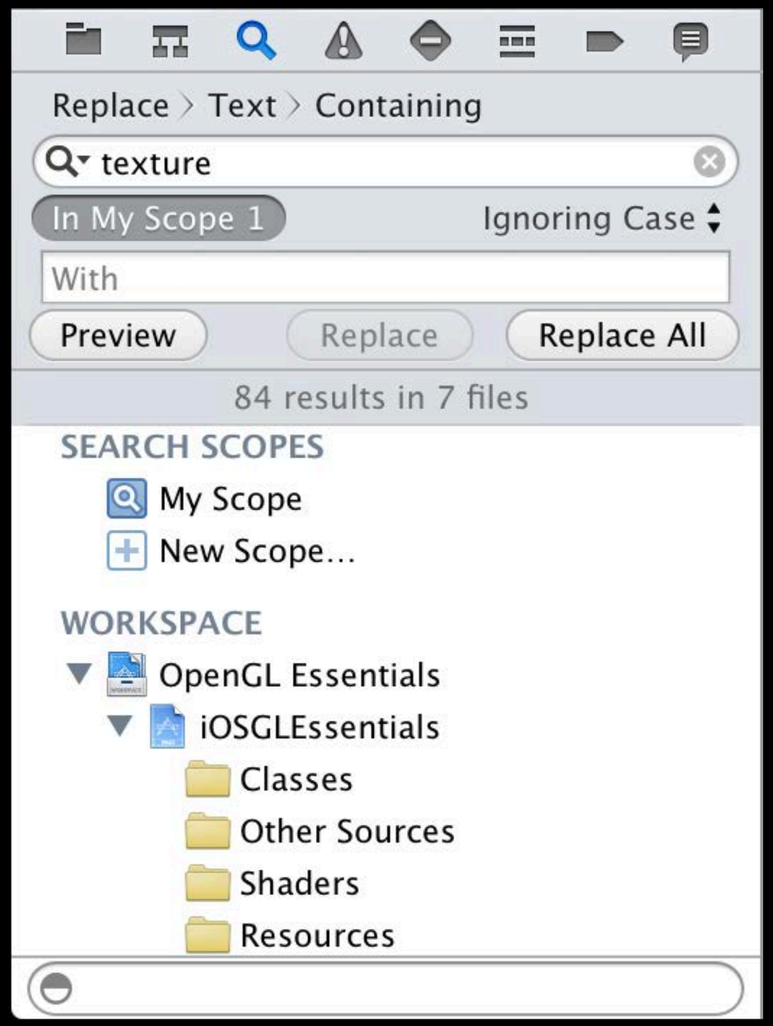


Find or replace plain text, regular expressions, etc.

Find Navigator

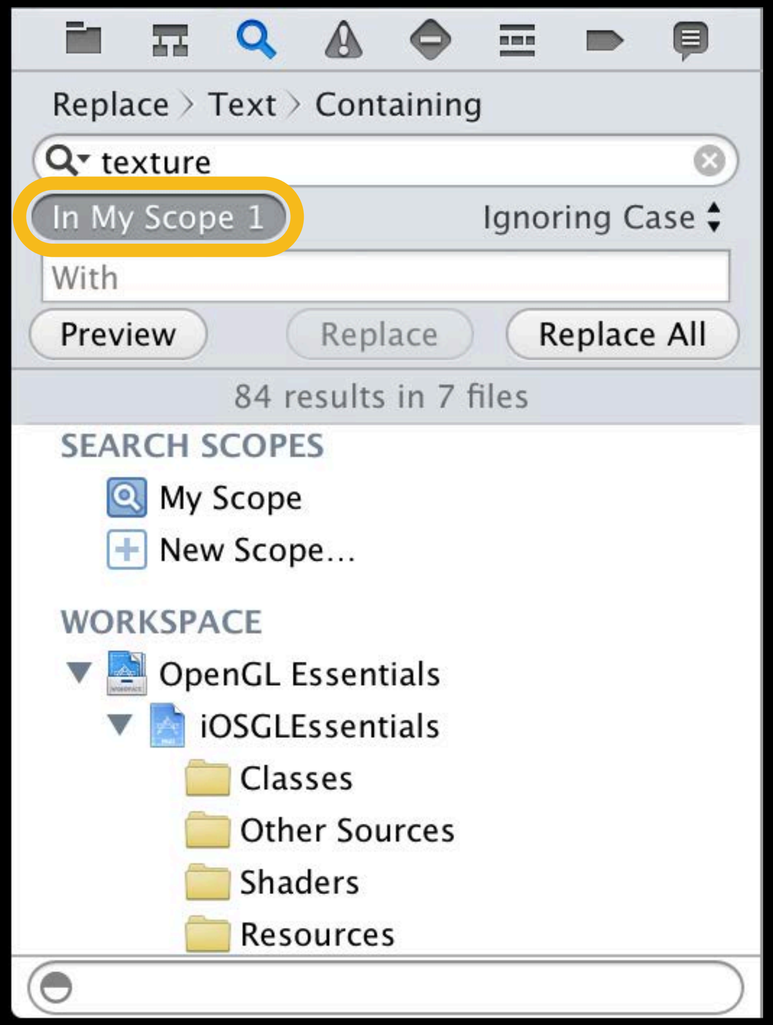
Find or replace plain text,
regular expressions, etc.

Find Navigator



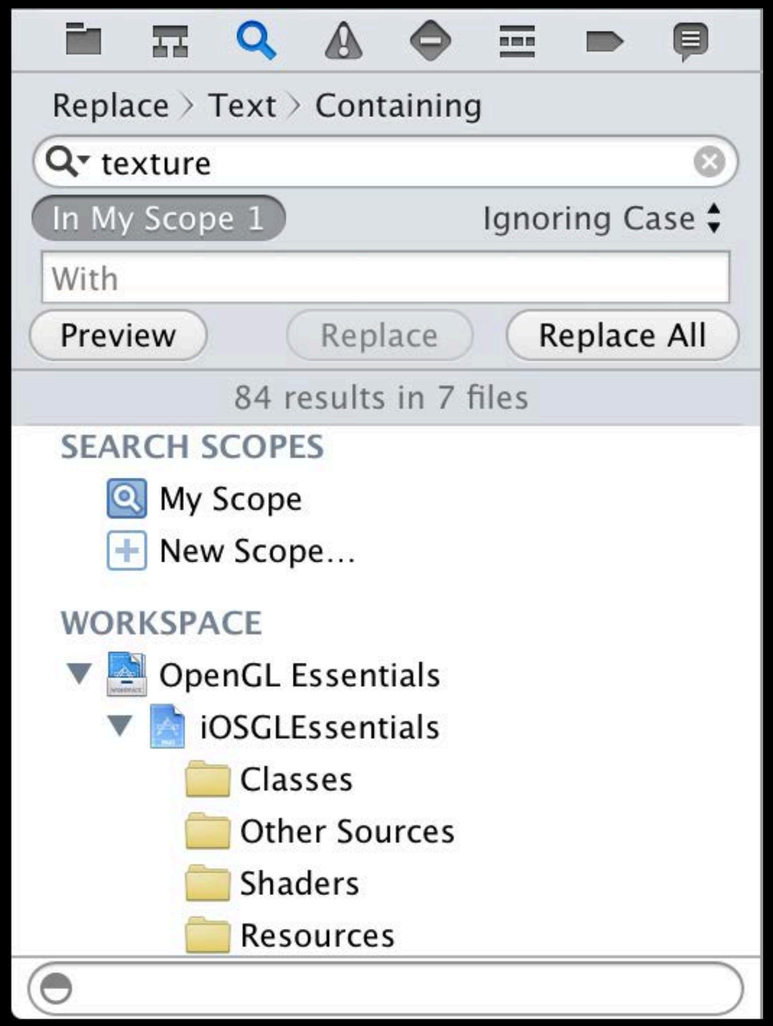
Find or replace plain text, regular expressions, etc.

Find Navigator



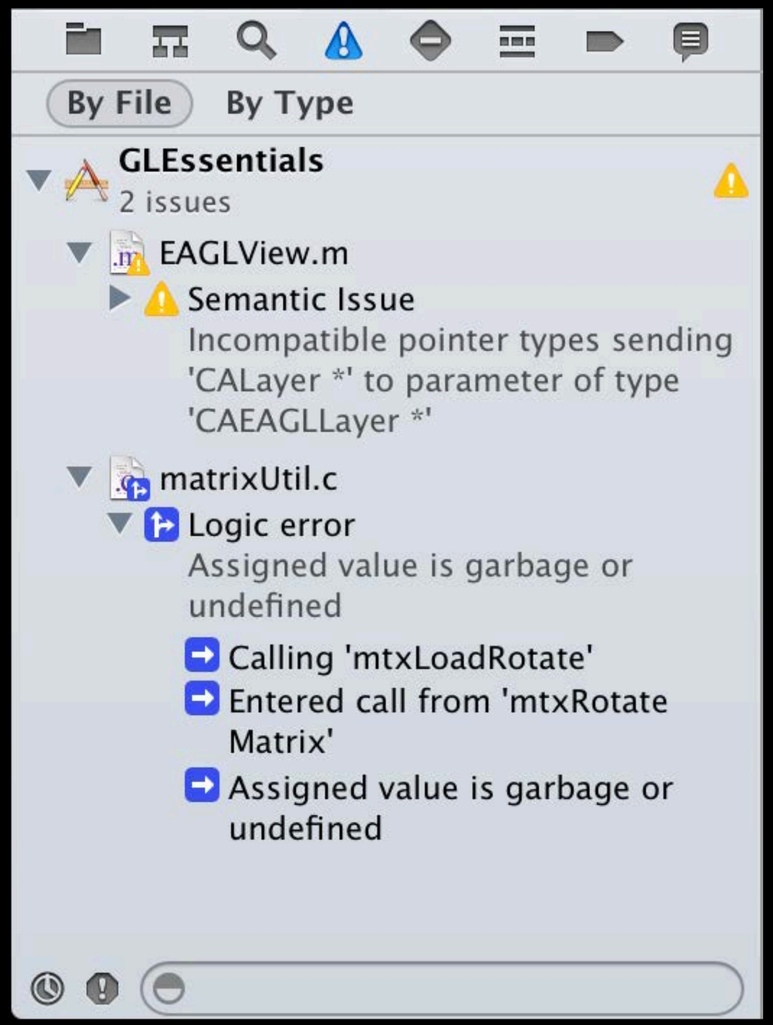
Find or replace plain text, regular expressions, etc.

Find Navigator



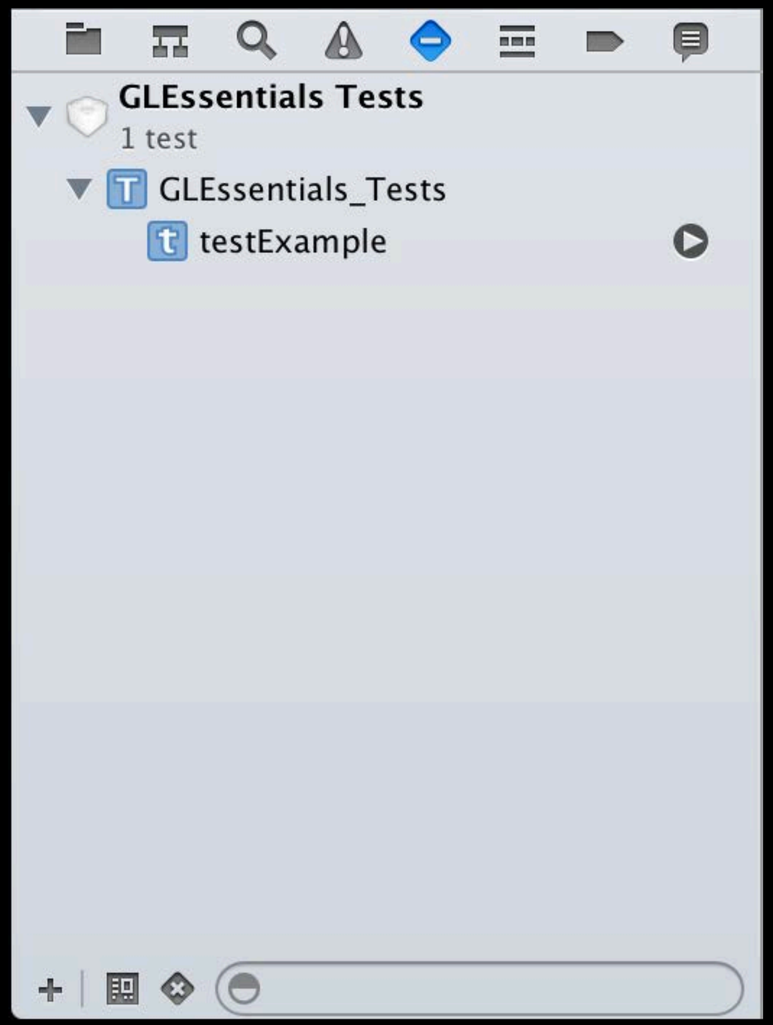
Find or replace plain text, regular expressions, etc.

Issue Navigator



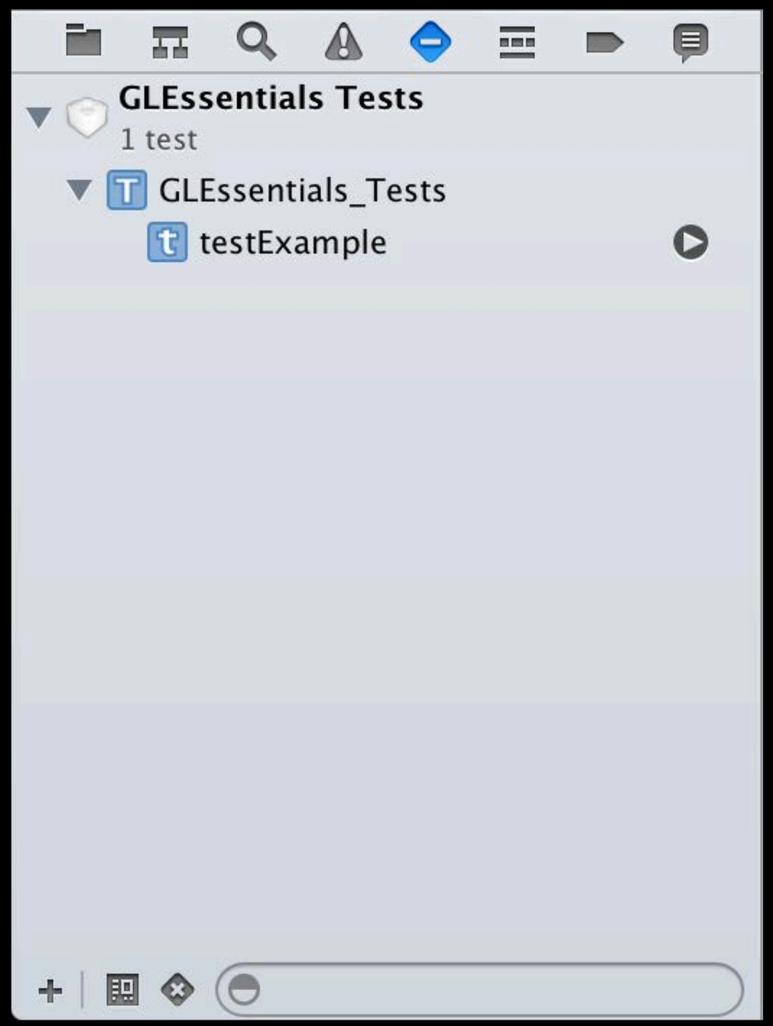
View and fix warnings, errors, and analysis issues

Test Navigator



Create, edit, and run unit tests for the products in your workspace

Test Navigator

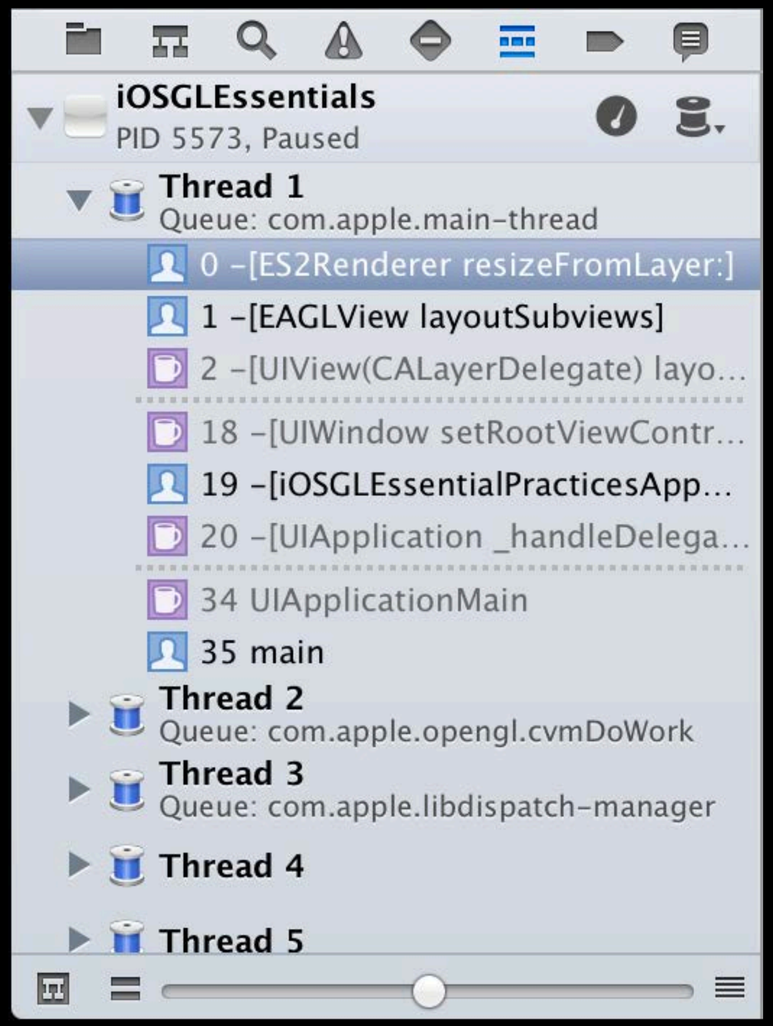


Create, edit, and run unit tests for the products in your workspace

Testing in Xcode 5

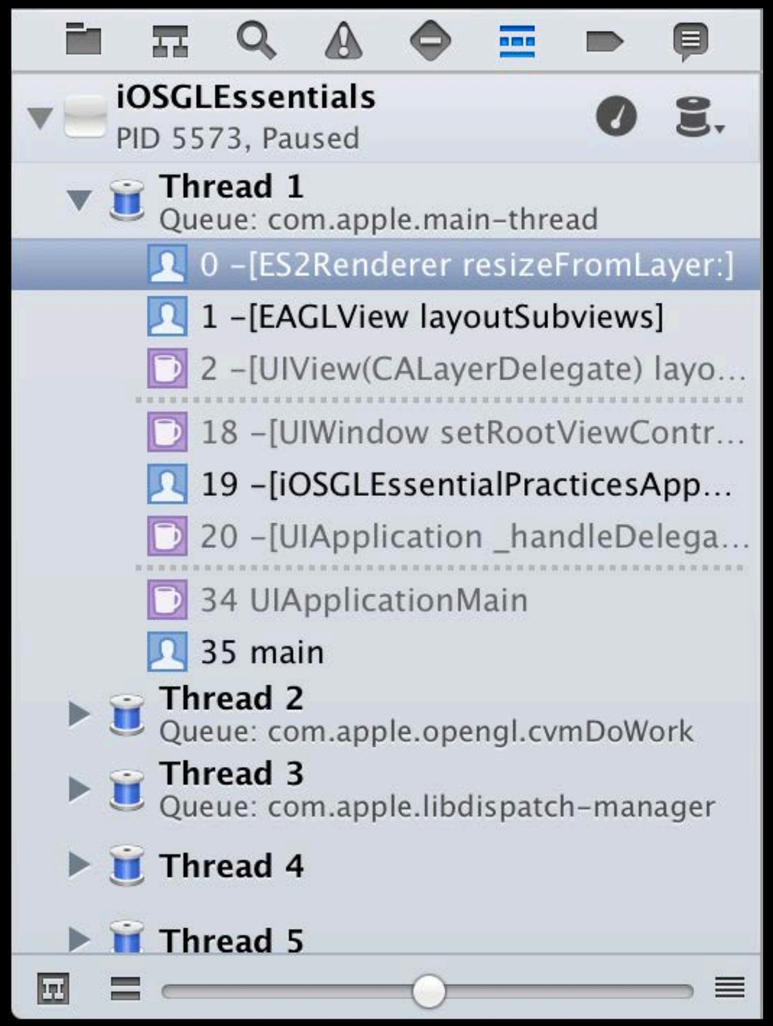
Pacific Heights
Thursday
9:00AM

Debug Navigator



Access backtraces, queues, and performance gauges

Debug Navigator

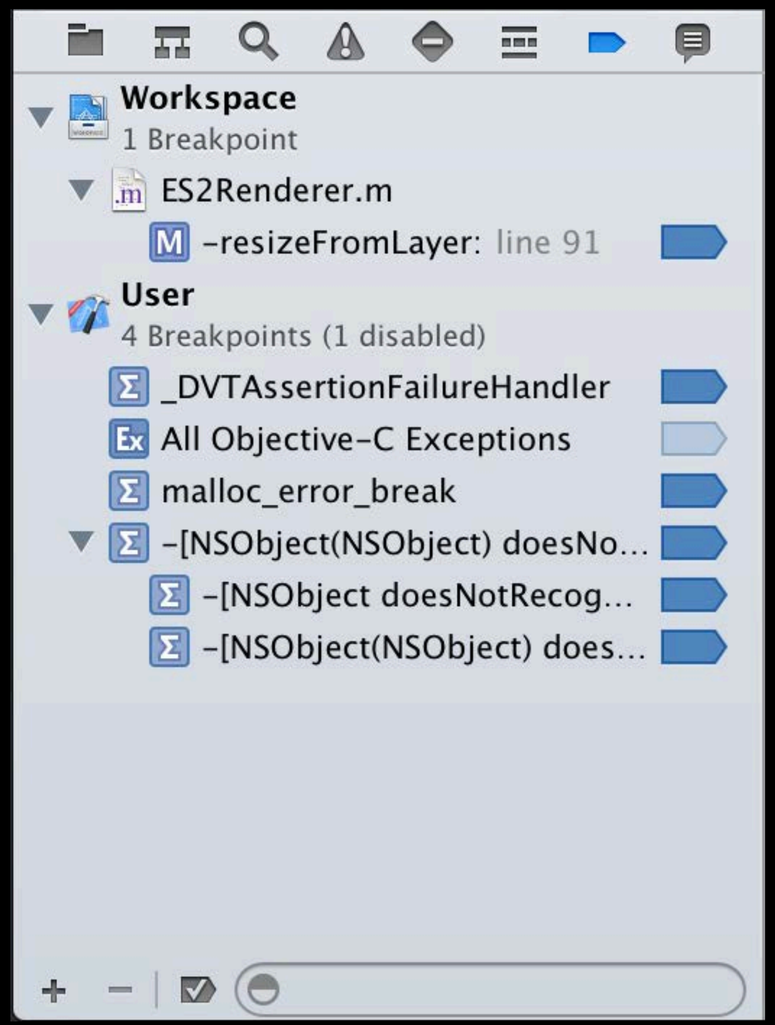


Access backtraces, queues, and performance gauges

Debugging with Xcode

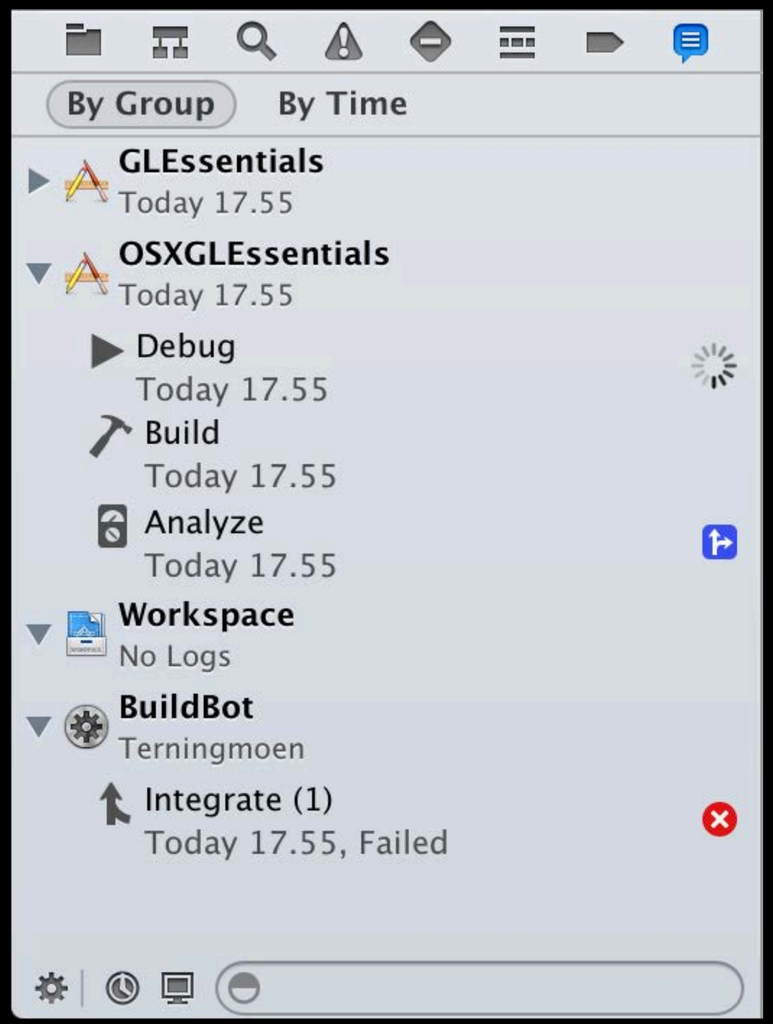
Pacific Heights
Wednesday
2:00PM

Breakpoint Navigator



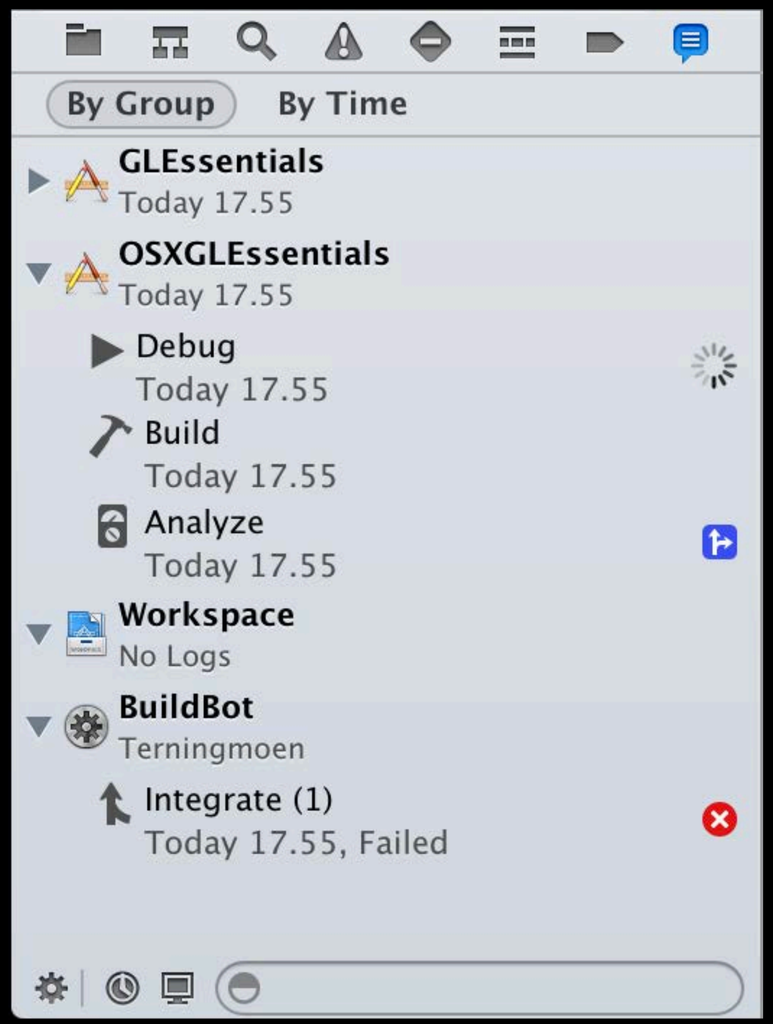
Create, edit, and control debugger breakpoints

Activity Log Navigator



View and search build logs, SCM logs, debug logs, etc

Activity Log Navigator



View and search build logs, SCM logs, debug logs, etc

Continuous Integration with Xcode 5

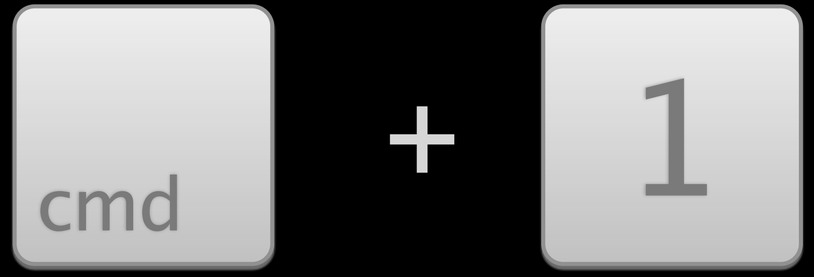
Presidio
Tuesday
3:15PM

Keyboard Access

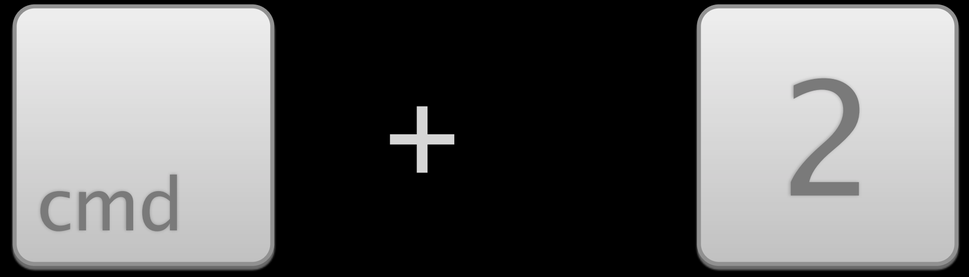
Keyboard Access

+

Keyboard Access



Keyboard Access



Keyboard Access

Keyboard Access

cmd + 4

Keyboard Access

cmd + 5

Keyboard Access

cmd + 6

Keyboard Access

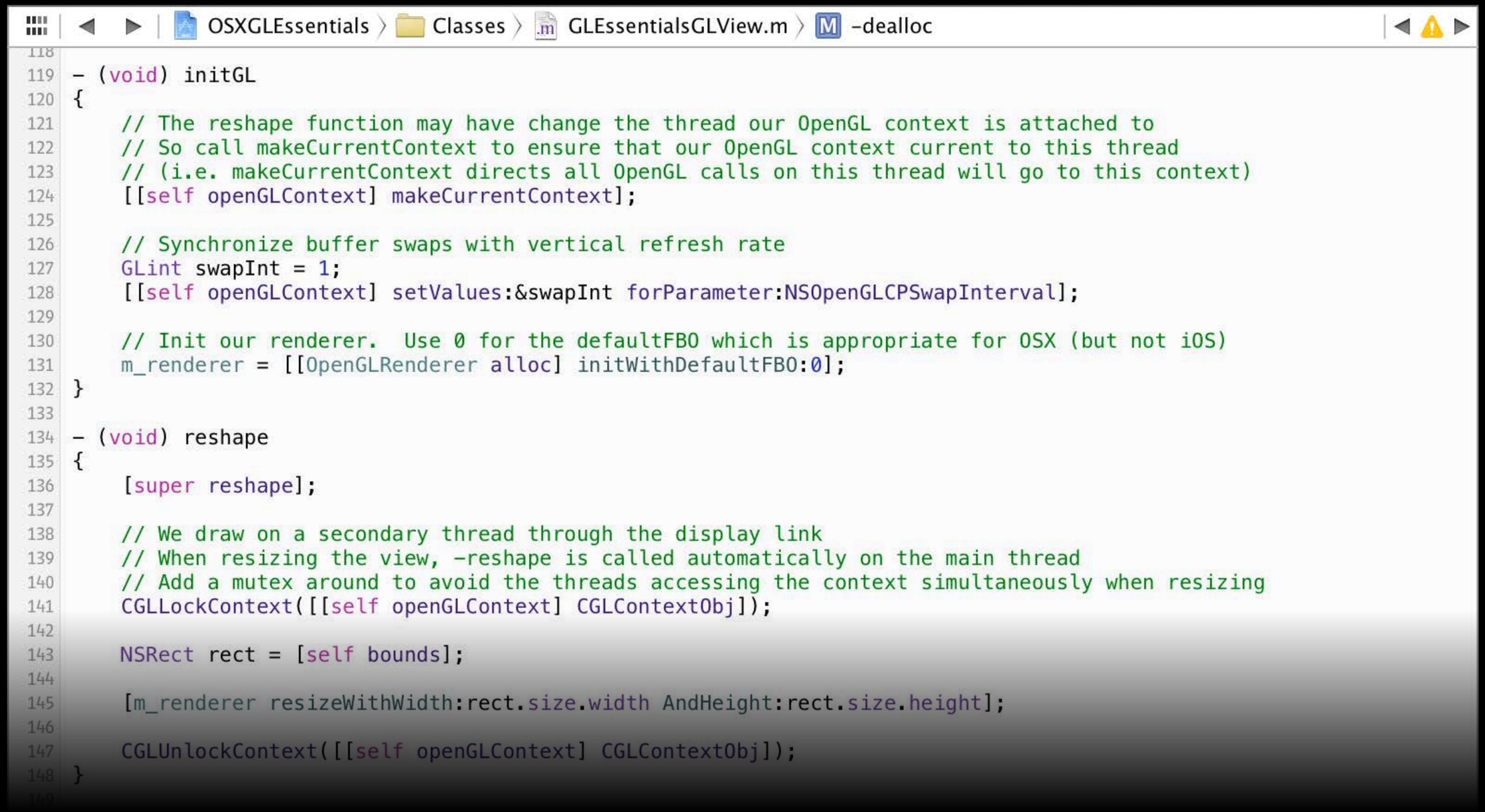
cmd + 7

Keyboard Access

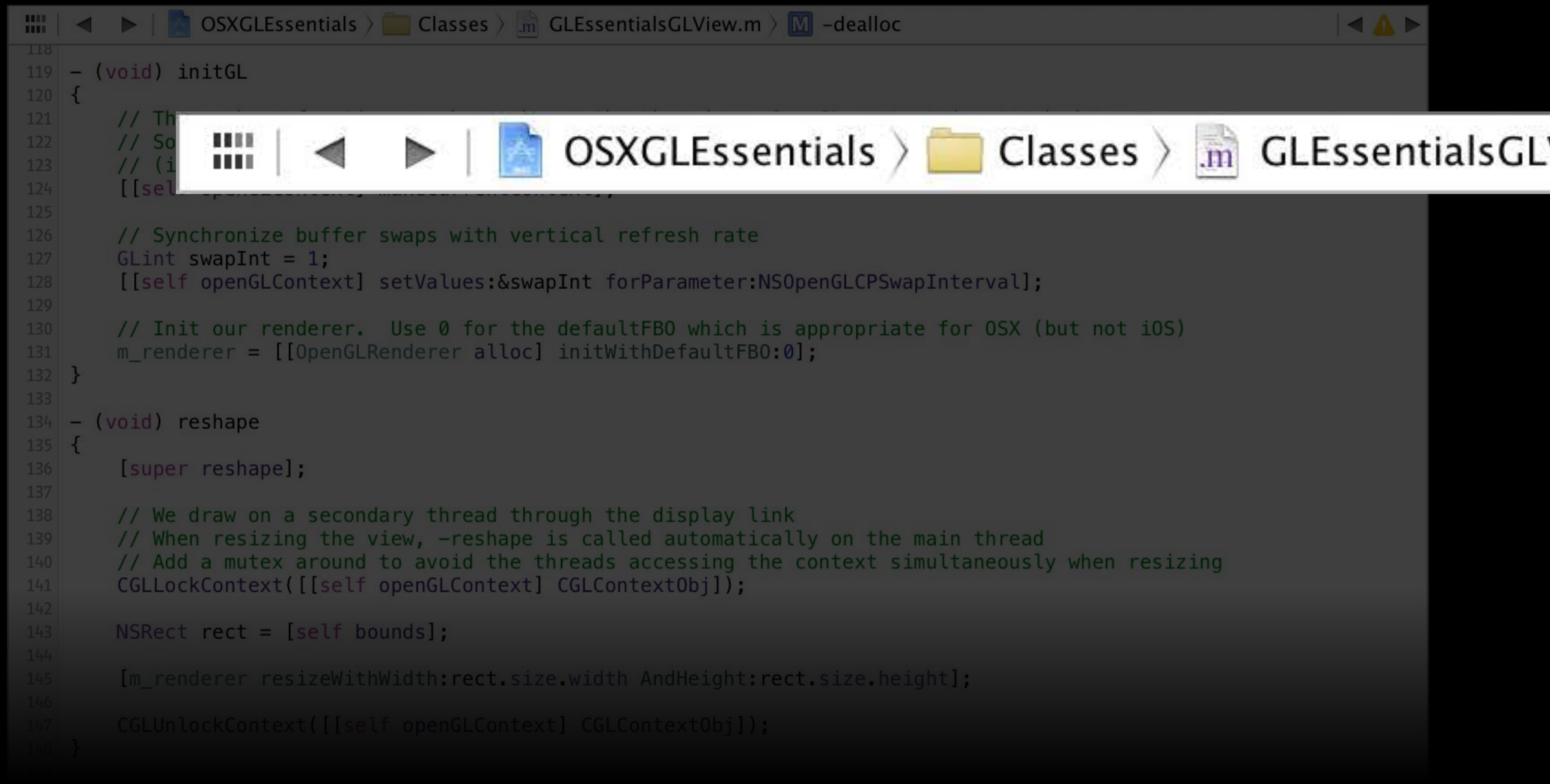
cmd + 8

Working Effectively with Editors

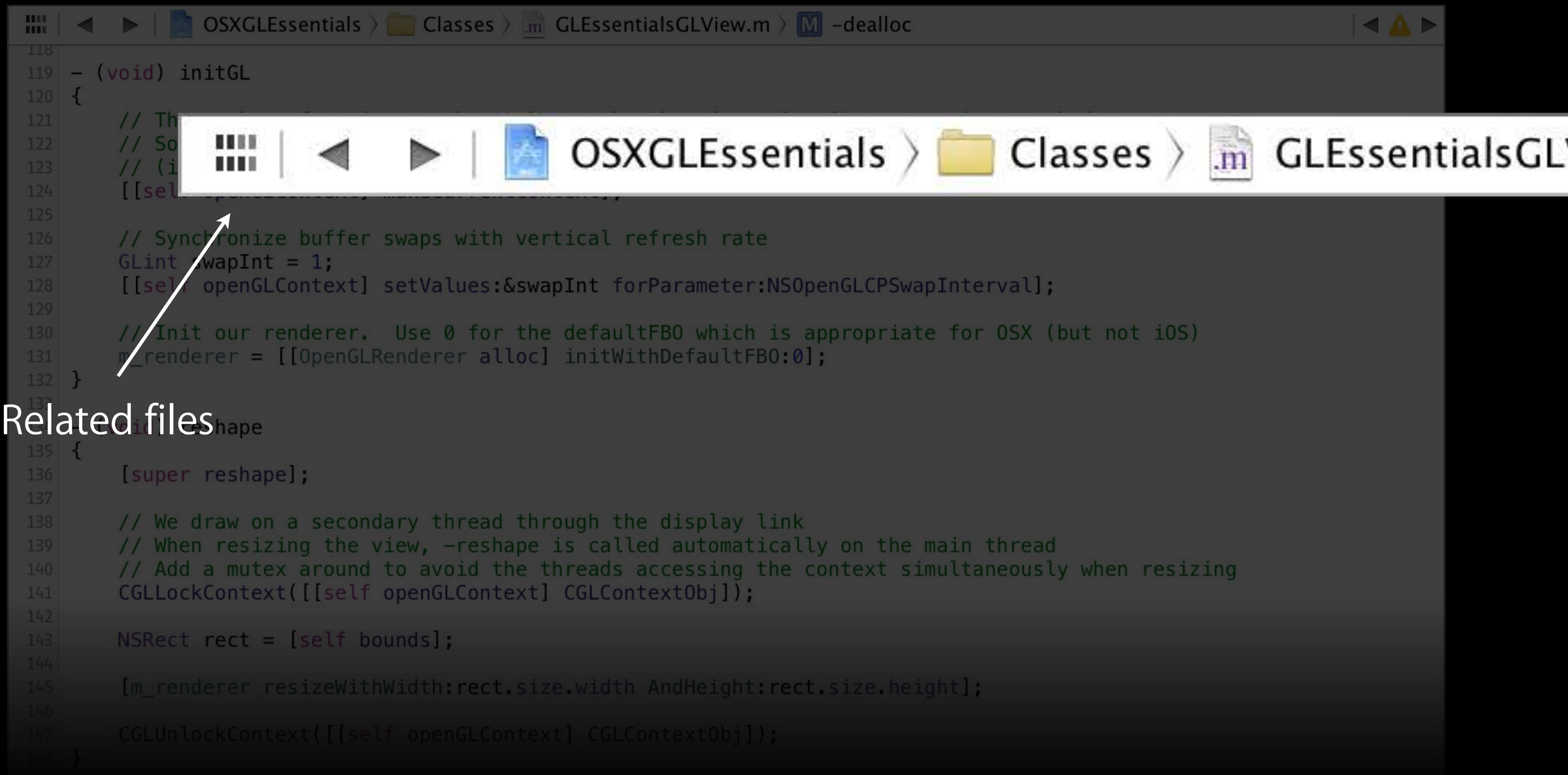
Editor Jump Bar



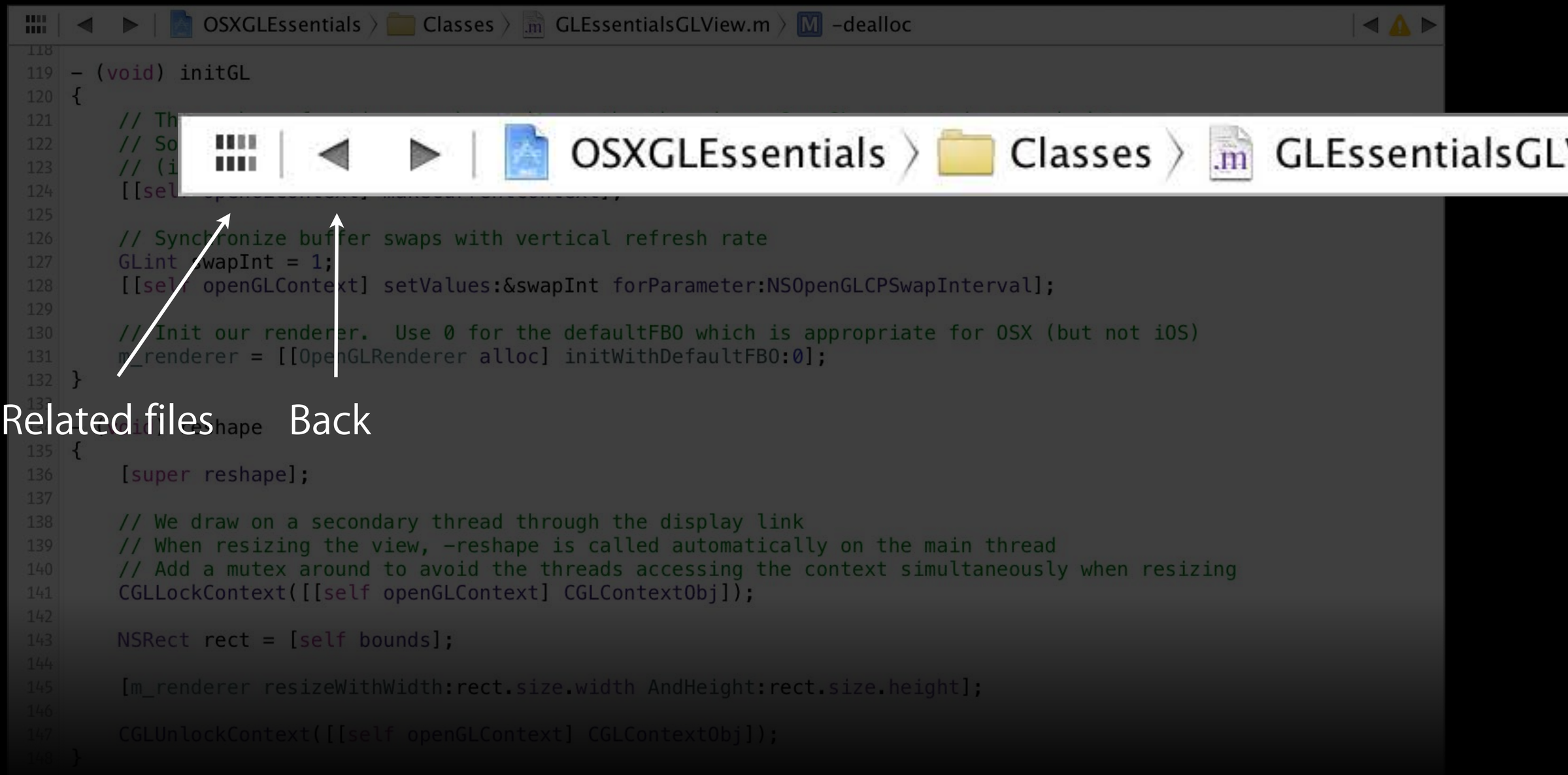
Editor Jump Bar



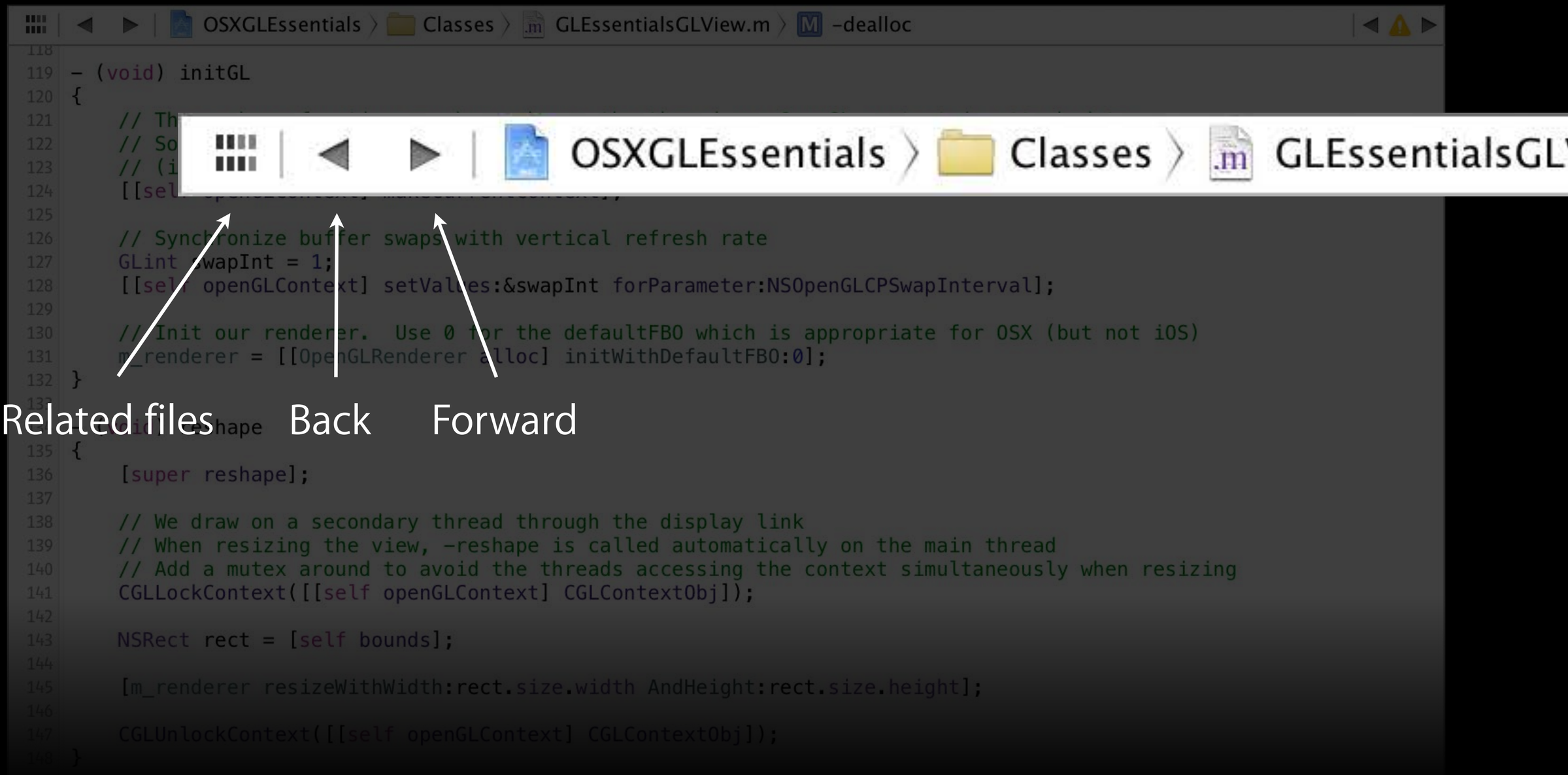
Editor Jump Bar



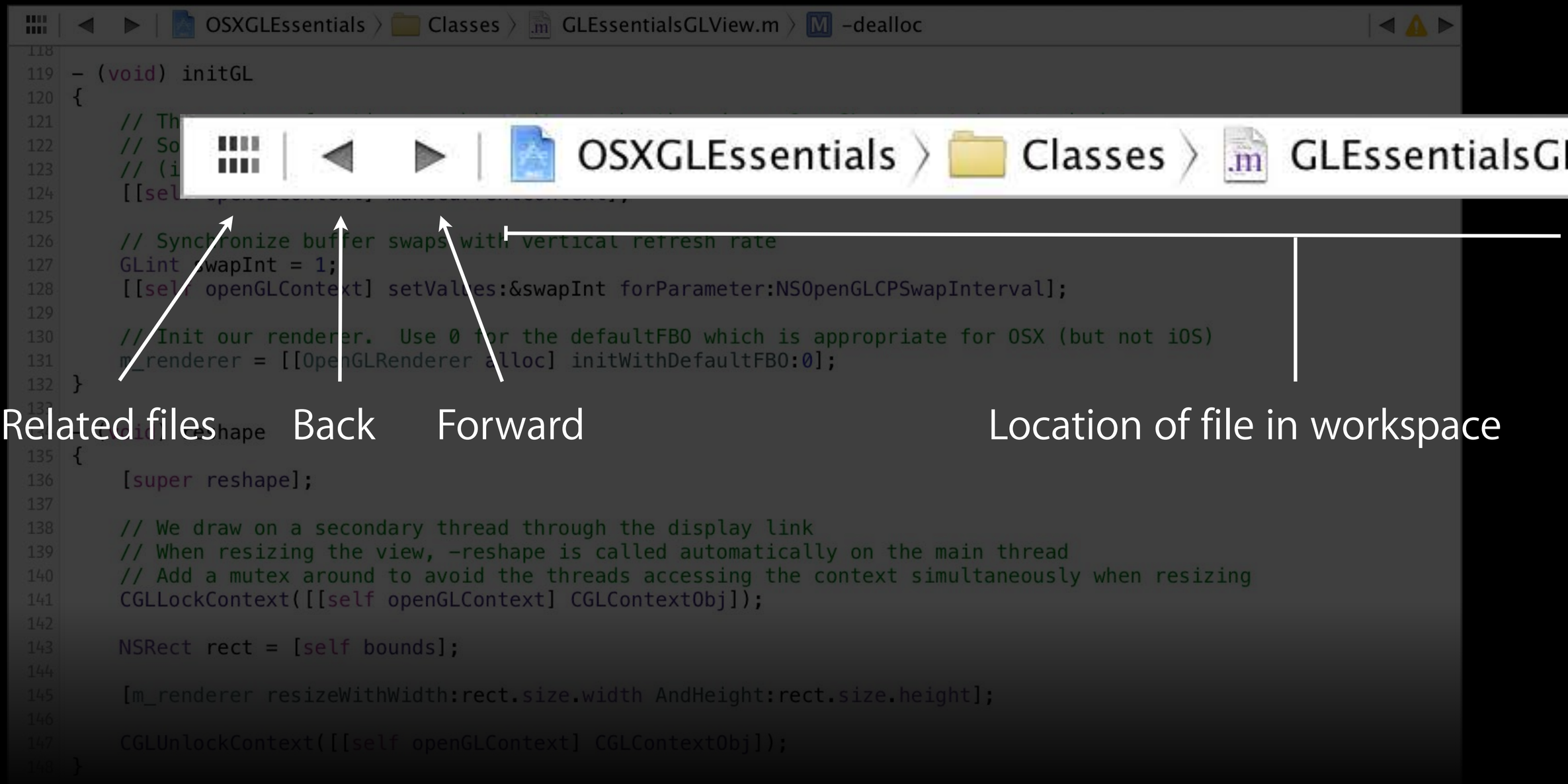
Editor Jump Bar



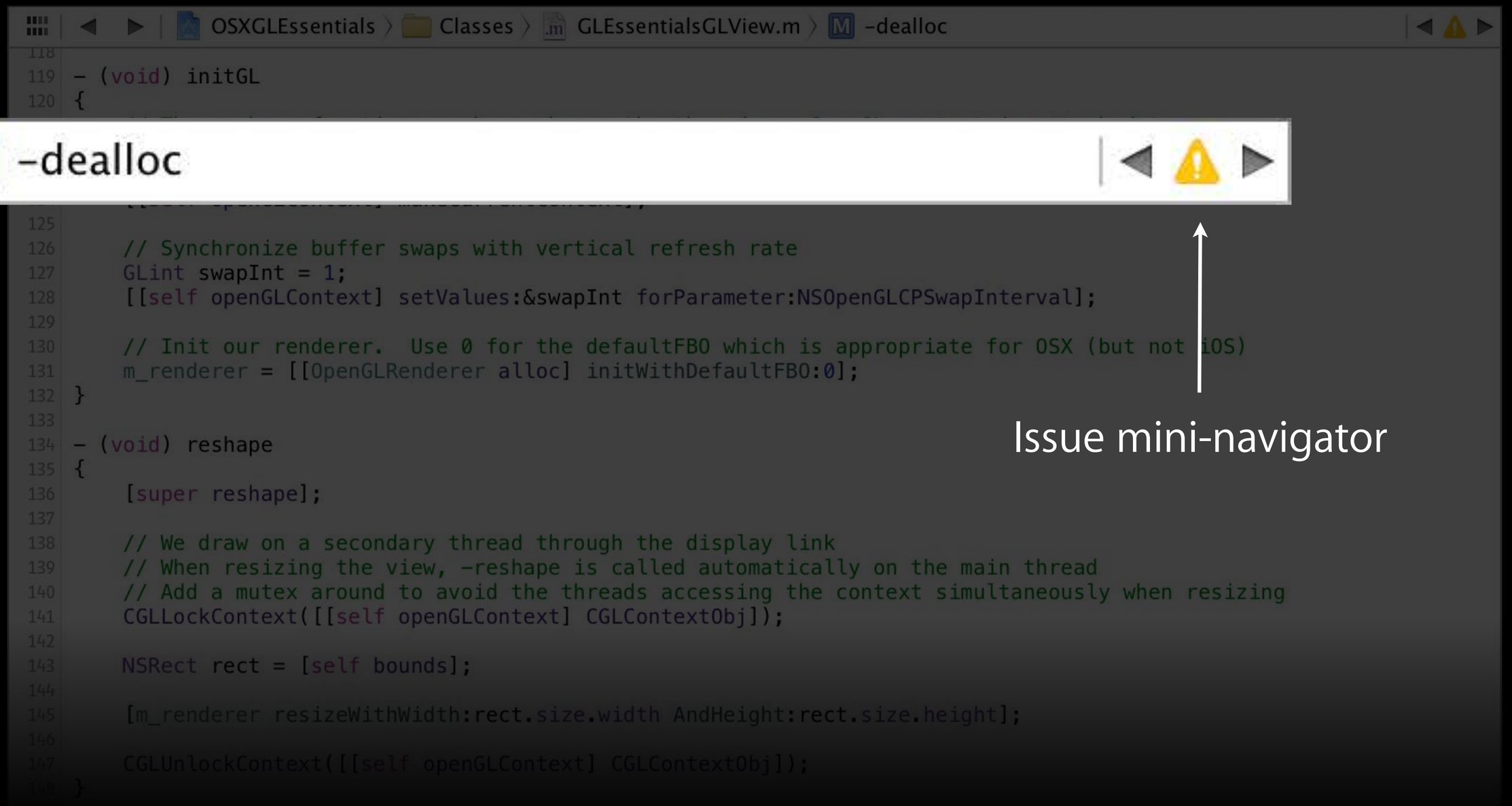
Editor Jump Bar



Editor Jump Bar

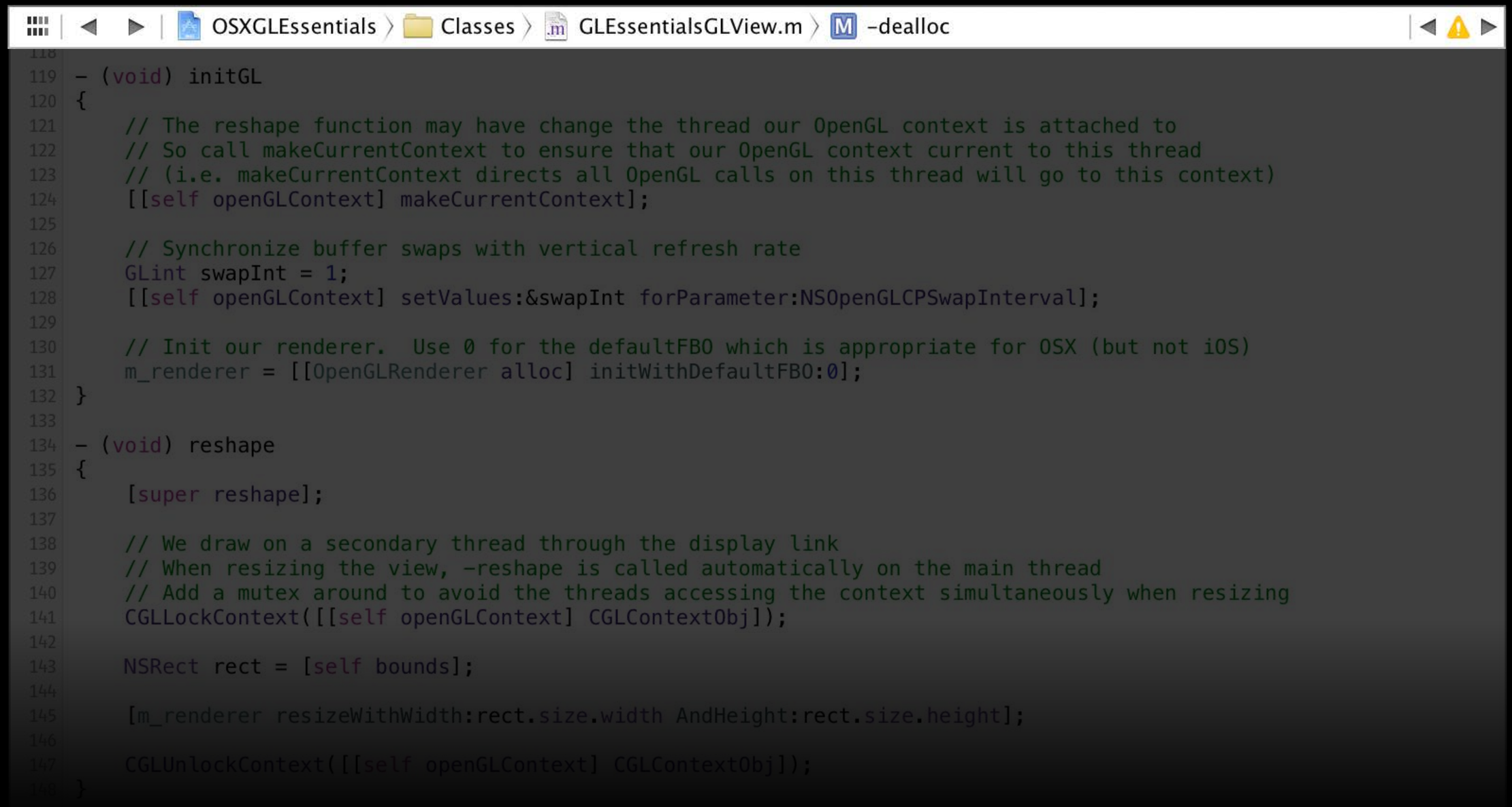


Editor Jump Bar



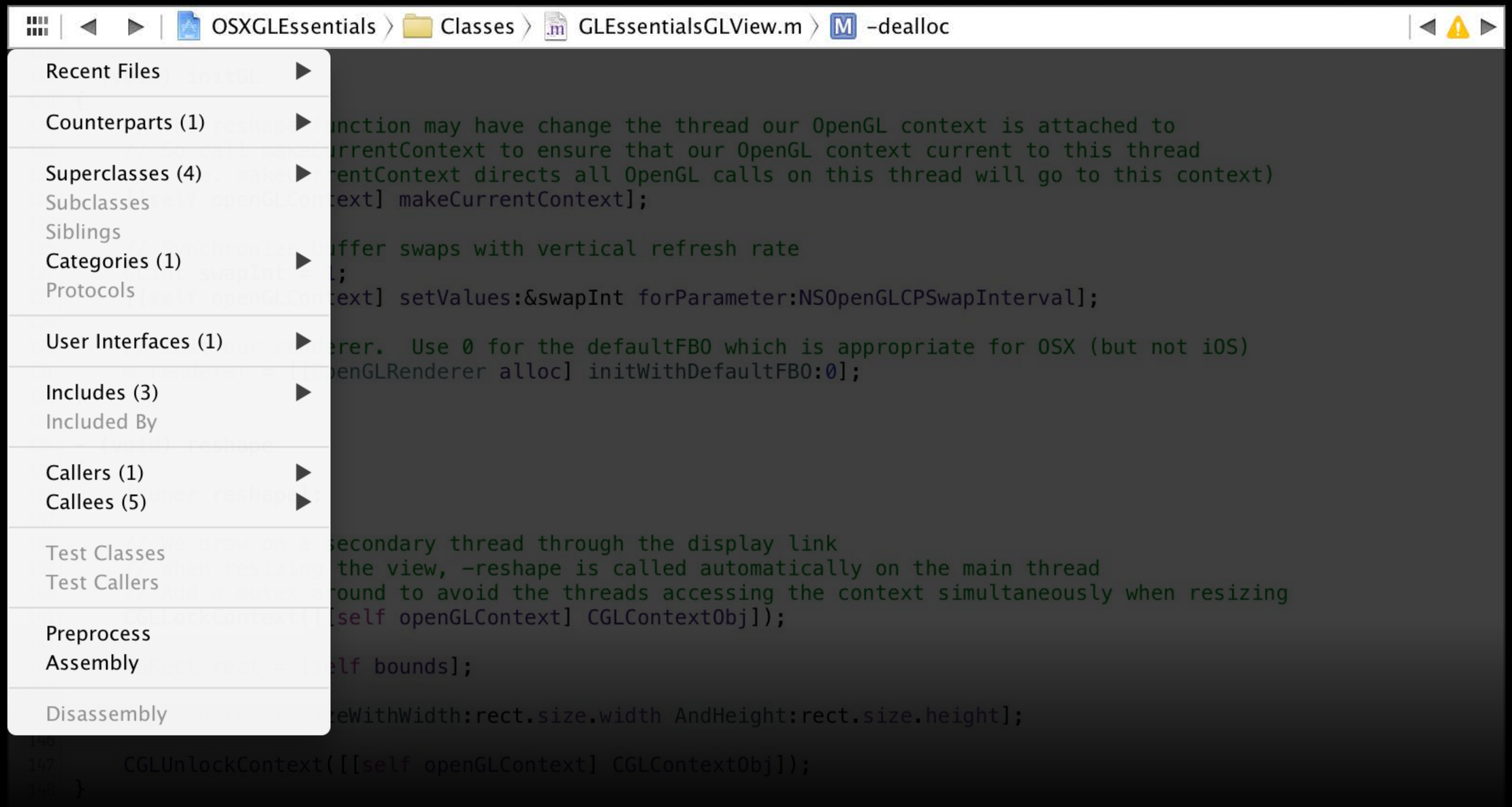
Issue mini-navigator

Editor Jump Bar

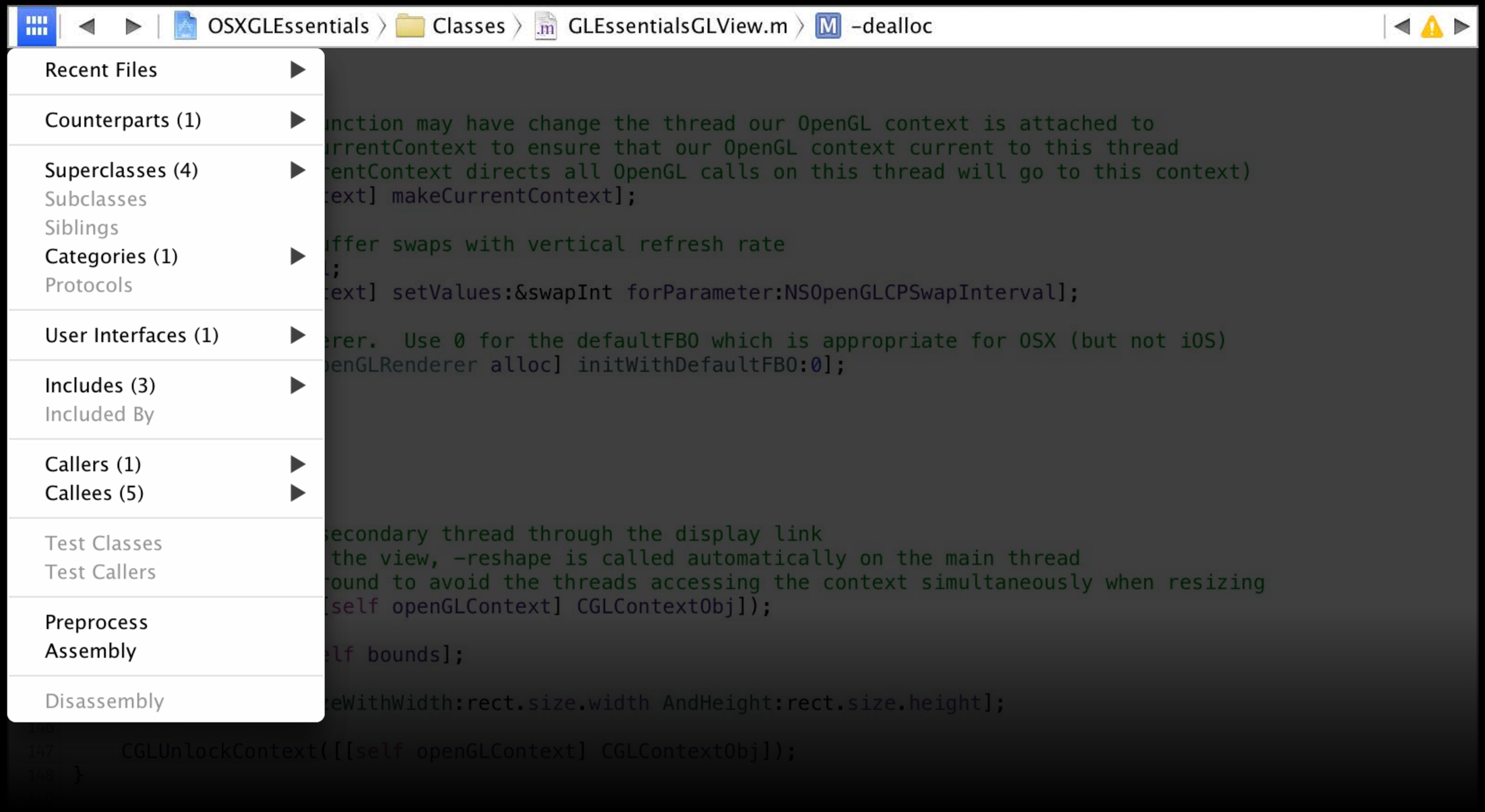



```
119 - (void) initGL
120 {
121     // The reshape function may have change the thread our OpenGL context is attached to
122     // So call makeCurrentContext to ensure that our OpenGL context current to this thread
123     // (i.e. makeCurrentContext directs all OpenGL calls on this thread will go to this context)
124     [[self openGLContext] makeCurrentContext];
125 
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129 
130     // Init our renderer.  Use 0 for the defaultFBO which is appropriate for OSX (but not iOS)
131     m_renderer = [[OpenGLRenderer alloc] initWithDefaultFBO:0];
132 }
133 
134 - (void) reshape
135 {
136     [super reshape];
137 
138     // We draw on a secondary thread through the display link
139     // When resizing the view, -reshape is called automatically on the main thread
140     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
141     CGLLockContext([[self openGLContext] CGLContextObj]);
142 
143     NSRect rect = [self bounds];
144 
145     [m_renderer resizeWithWidth:rect.size.width AndHeight:rect.size.height];
146 
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
```

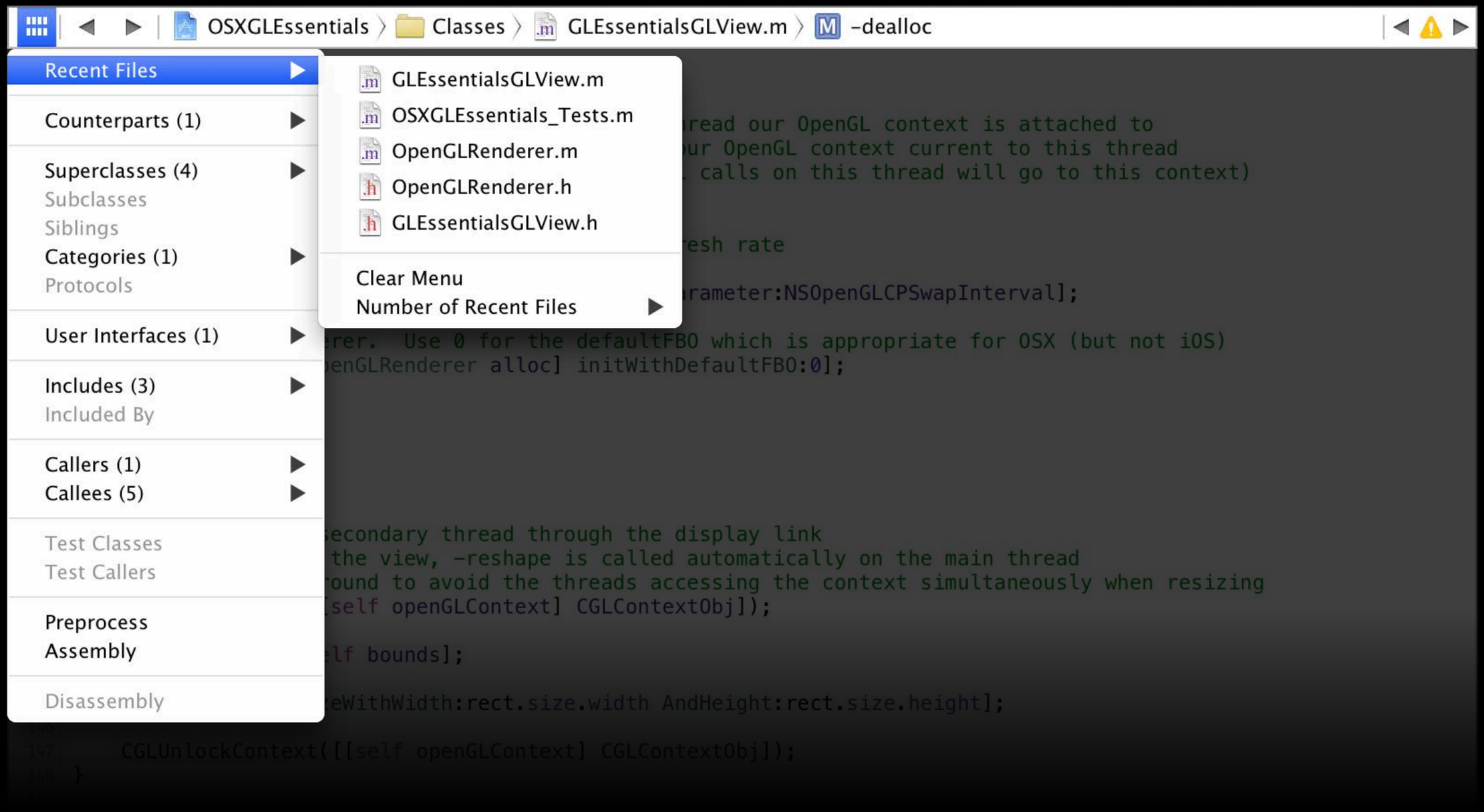
Editor Jump Bar



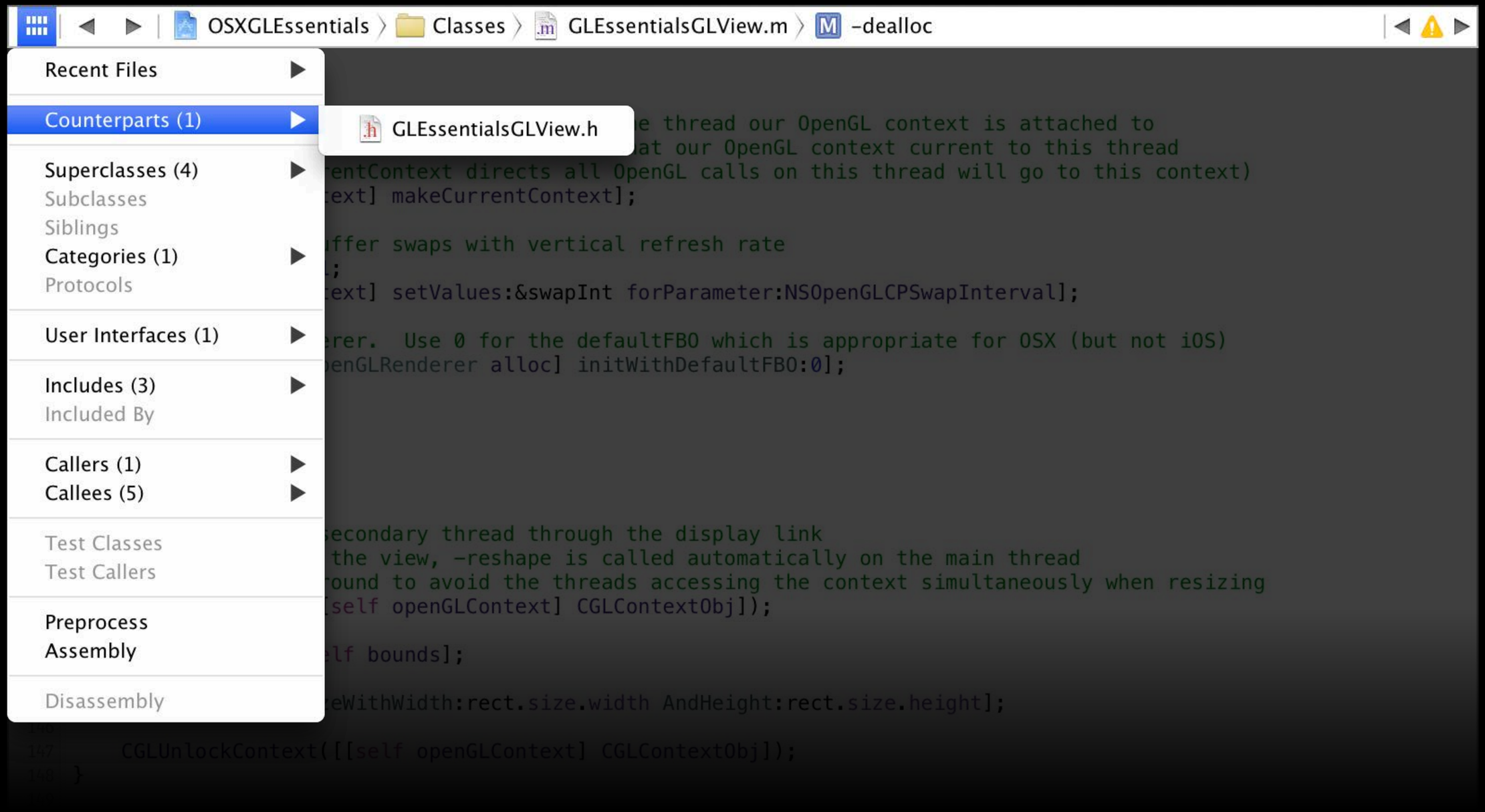
Editor Jump Bar



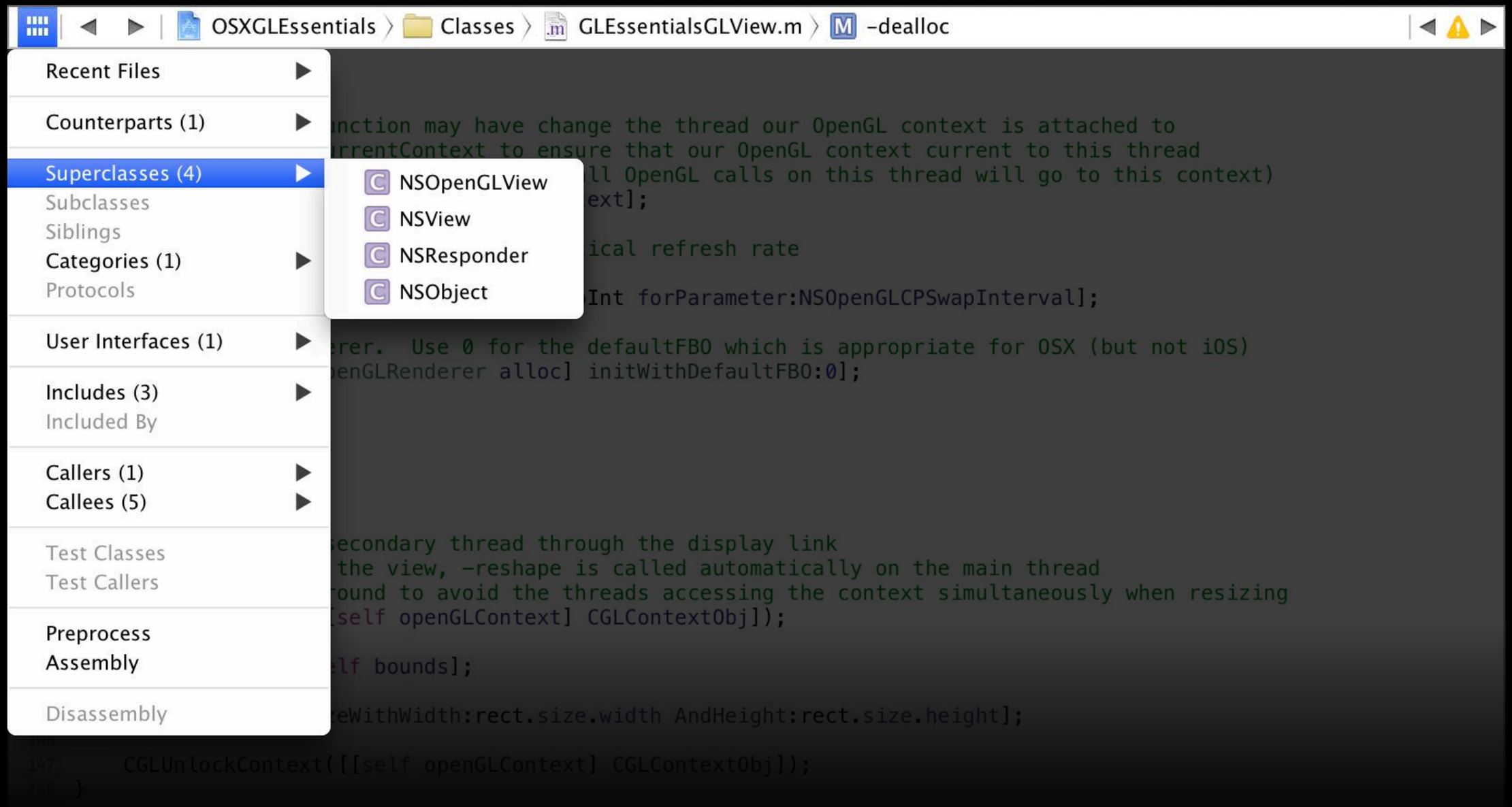
Editor Jump Bar



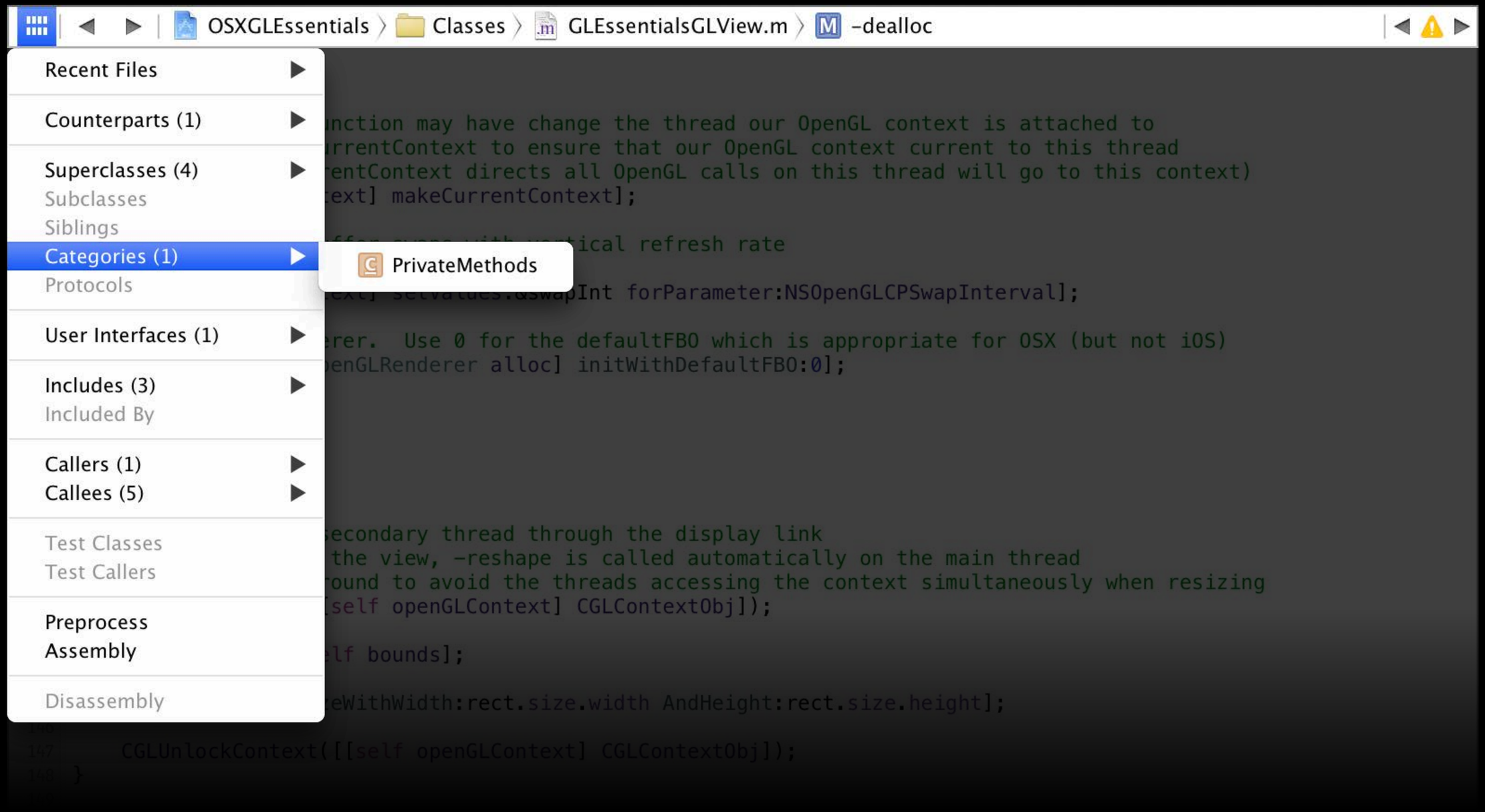
Editor Jump Bar



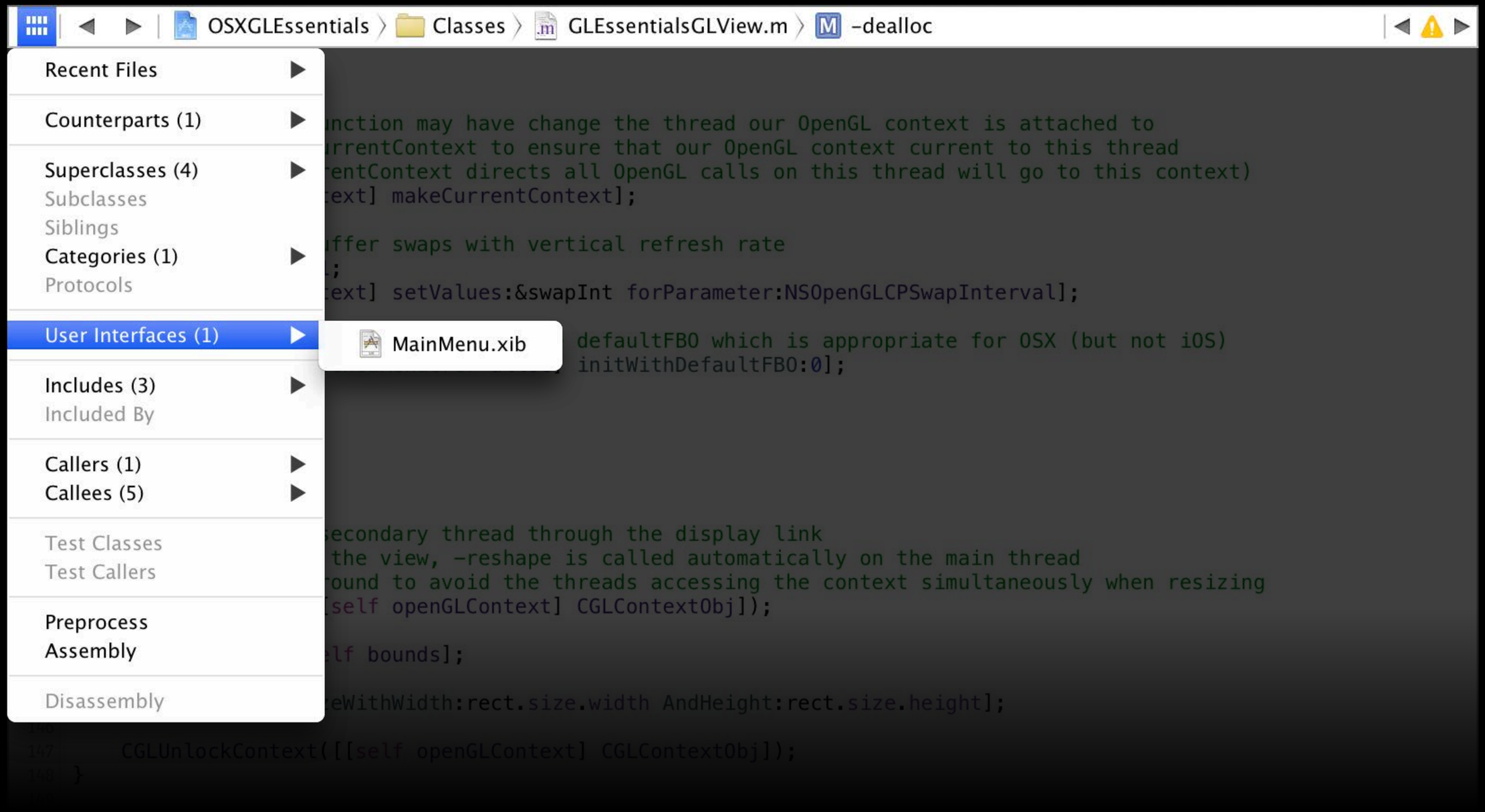
Editor Jump Bar



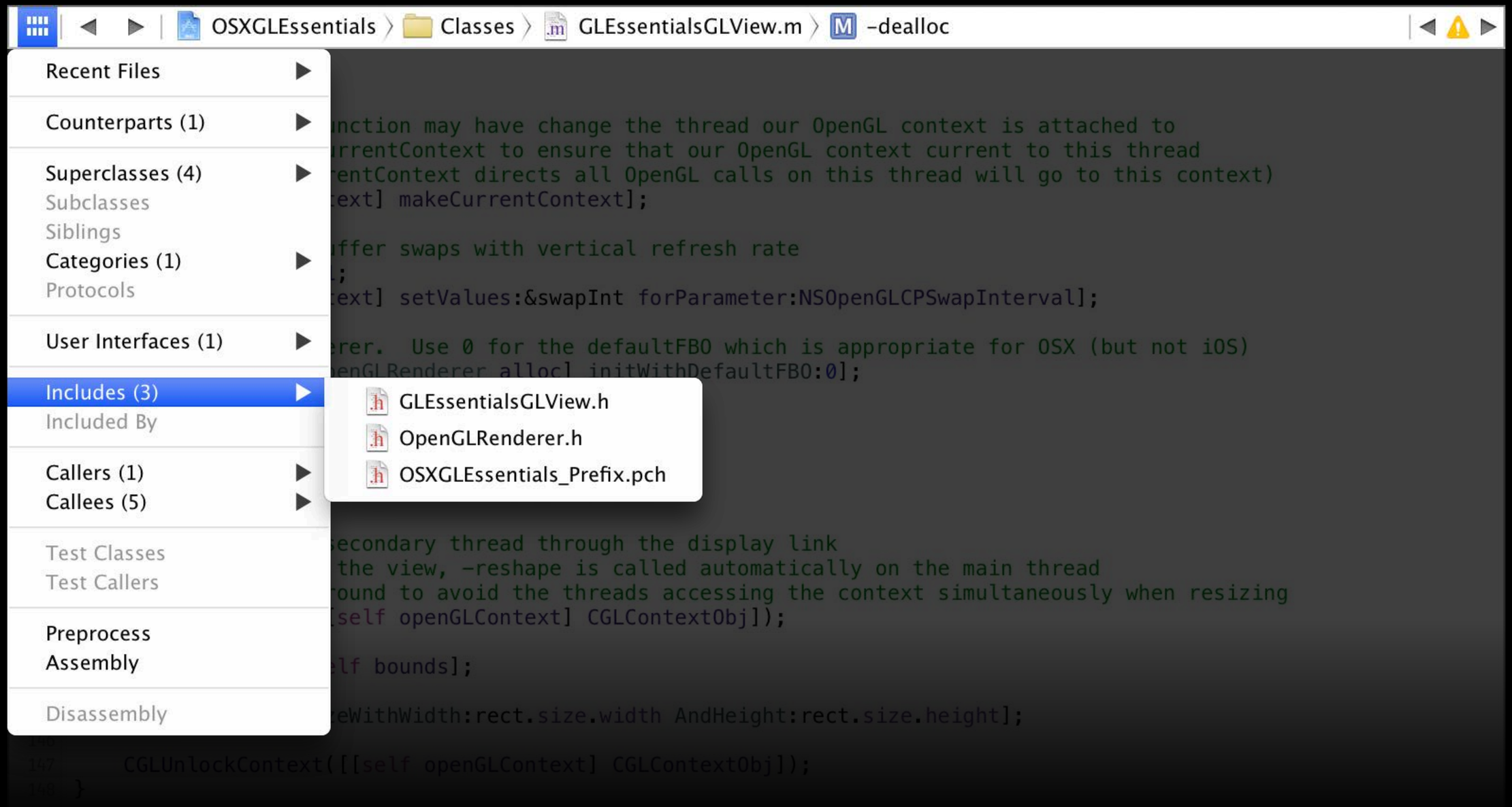
Editor Jump Bar



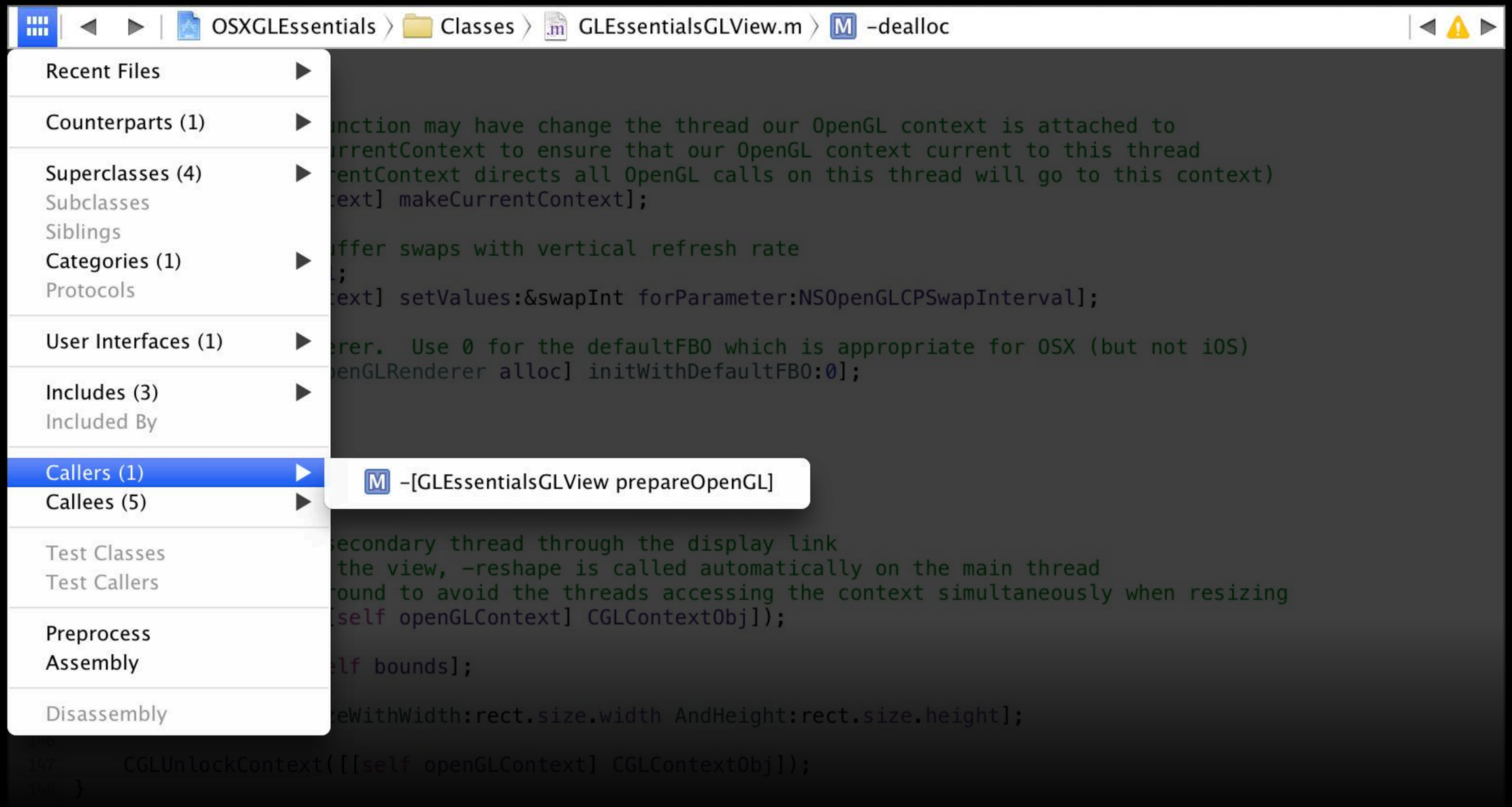
Editor Jump Bar



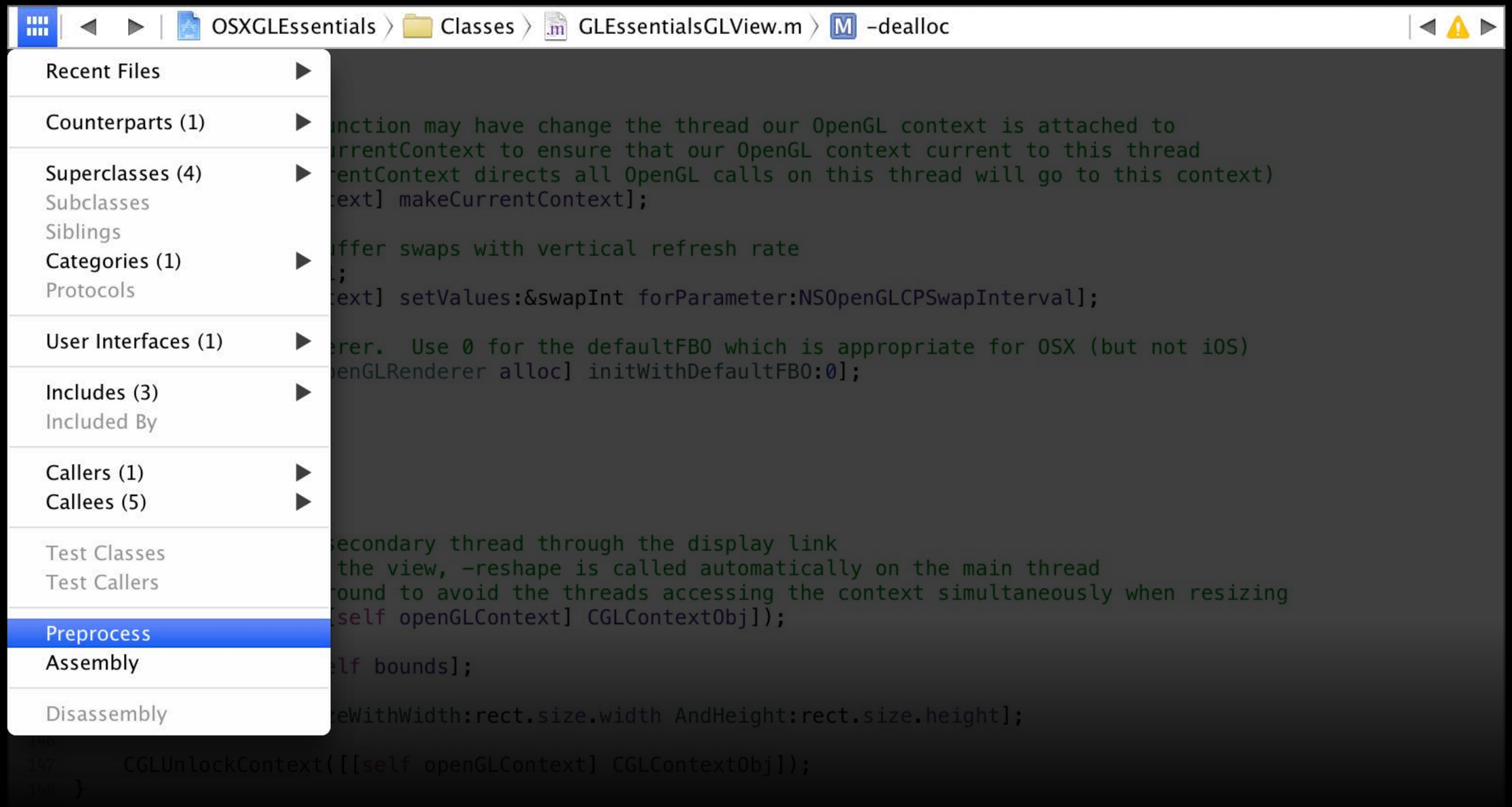
Editor Jump Bar



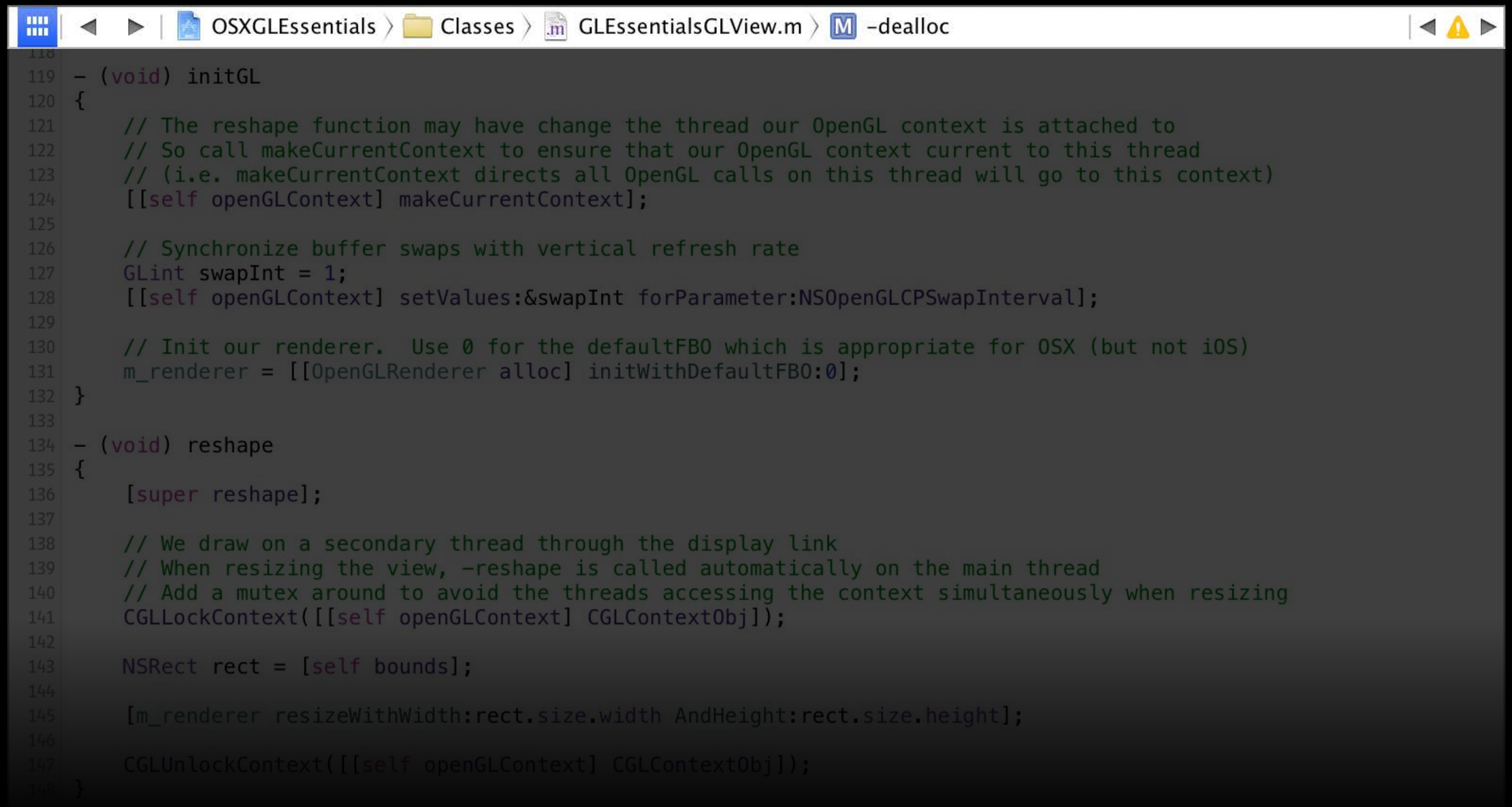
Editor Jump Bar



Editor Jump Bar

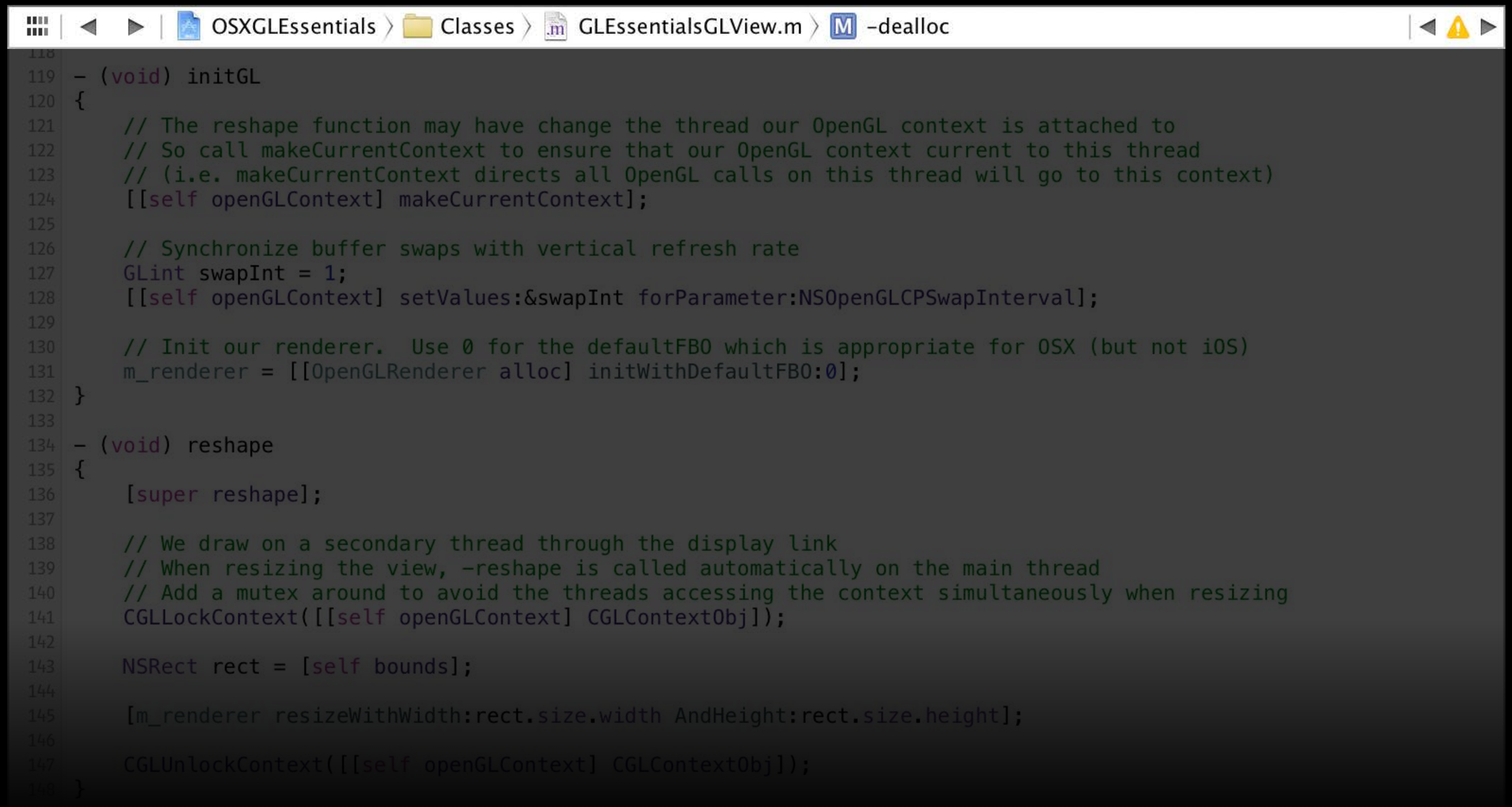


Editor Jump Bar



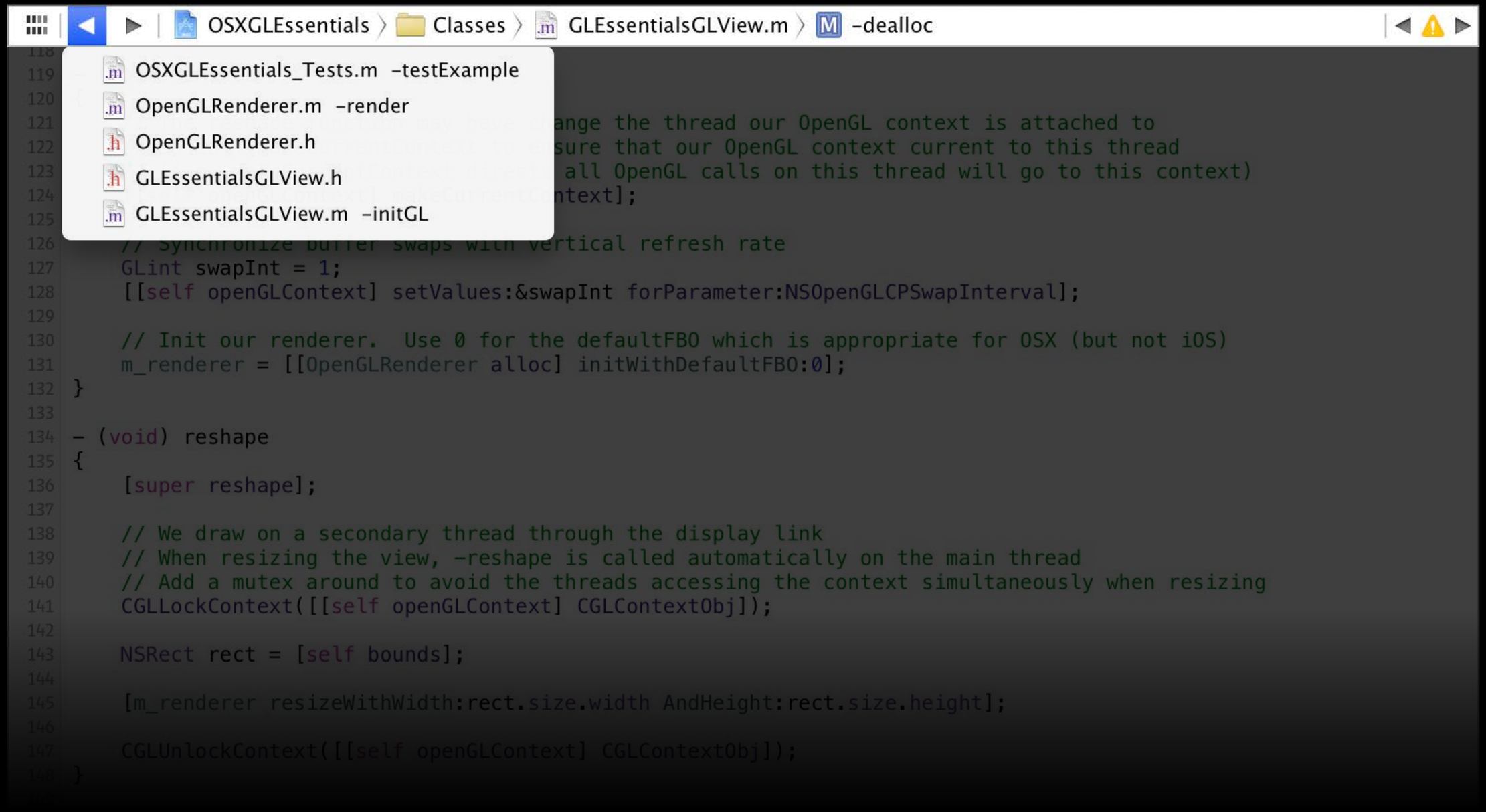

```
119 - (void) initGL
120 {
121     // The reshape function may have change the thread our OpenGL context is attached to
122     // So call makeCurrentContext to ensure that our OpenGL context current to this thread
123     // (i.e. makeCurrentContext directs all OpenGL calls on this thread will go to this context)
124     [[self openGLContext] makeCurrentContext];
125 
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129 
130     // Init our renderer.  Use 0 for the defaultFBO which is appropriate for OSX (but not iOS)
131     m_renderer = [[OpenGLRenderer alloc] initWithDefaultFBO:0];
132 }
133 
134 - (void) reshape
135 {
136     [super reshape];
137 
138     // We draw on a secondary thread through the display link
139     // When resizing the view, -reshape is called automatically on the main thread
140     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
141     CGLLockContext([[self openGLContext] CGLContextObj]);
142 
143     NSRect rect = [self bounds];
144 
145     [m_renderer resizeWithWidth:rect.size.width AndHeight:rect.size.height];
146 
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
```

Editor Jump Bar

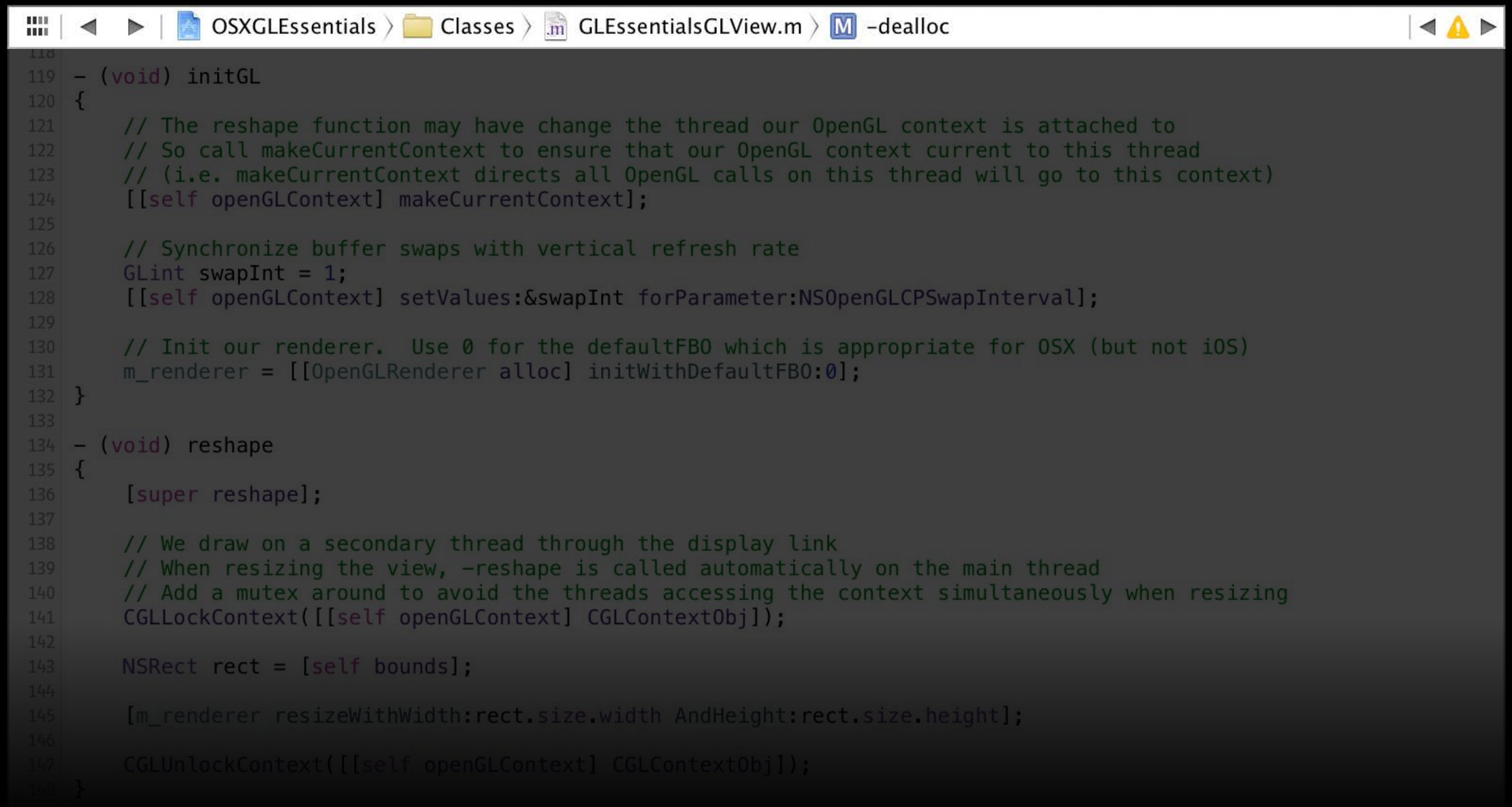



```
119 - (void) initGL
120 {
121     // The reshape function may have change the thread our OpenGL context is attached to
122     // So call makeCurrentContext to ensure that our OpenGL context current to this thread
123     // (i.e. makeCurrentContext directs all OpenGL calls on this thread will go to this context)
124     [[self openGLContext] makeCurrentContext];
125 
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129 
130     // Init our renderer.  Use 0 for the defaultFBO which is appropriate for OSX (but not iOS)
131     m_renderer = [[OpenGLRenderer alloc] initWithDefaultFBO:0];
132 }
133 
134 - (void) reshape
135 {
136     [super reshape];
137 
138     // We draw on a secondary thread through the display link
139     // When resizing the view, -reshape is called automatically on the main thread
140     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
141     CGLLockContext([[self openGLContext] CGLContextObj]);
142 
143     NSRect rect = [self bounds];
144 
145     [m_renderer resizeWithWidth:rect.size.width AndHeight:rect.size.height];
146 
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
```

Editor Jump Bar

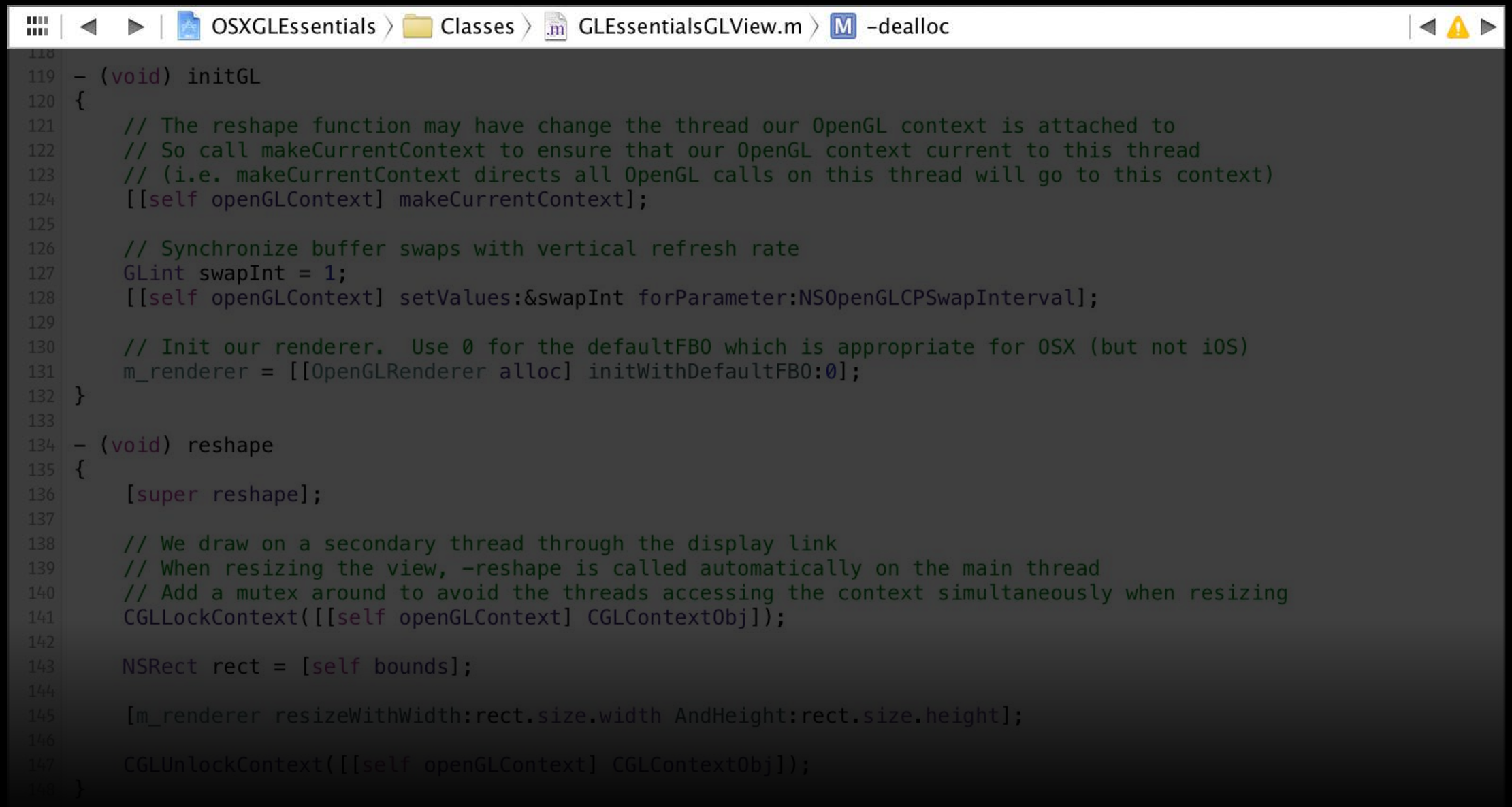


Editor Jump Bar



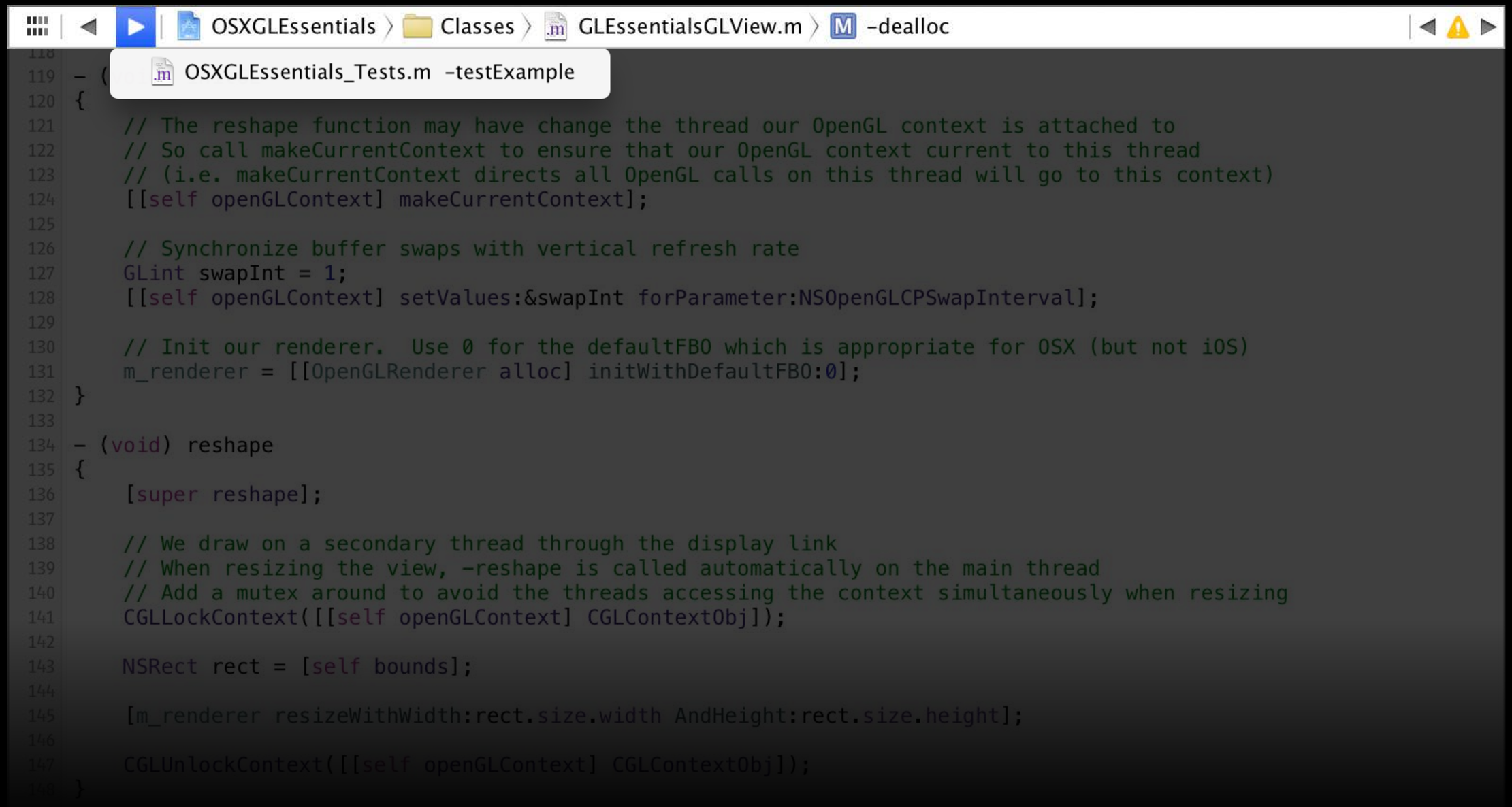

```
119 - (void) initGL
120 {
121     // The reshape function may have change the thread our OpenGL context is attached to
122     // So call makeCurrentContext to ensure that our OpenGL context current to this thread
123     // (i.e. makeCurrentContext directs all OpenGL calls on this thread will go to this context)
124     [[self openGLContext] makeCurrentContext];
125 
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129 
130     // Init our renderer.  Use 0 for the defaultFBO which is appropriate for OSX (but not iOS)
131     m_renderer = [[OpenGLRenderer alloc] initWithDefaultFBO:0];
132 }
133 
134 - (void) reshape
135 {
136     [super reshape];
137 
138     // We draw on a secondary thread through the display link
139     // When resizing the view, -reshape is called automatically on the main thread
140     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
141     CGLLockContext([[self openGLContext] CGLContextObj]);
142 
143     NSRect rect = [self bounds];
144 
145     [m_renderer resizeWithWidth:rect.size.width AndHeight:rect.size.height];
146 
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
```

Editor Jump Bar

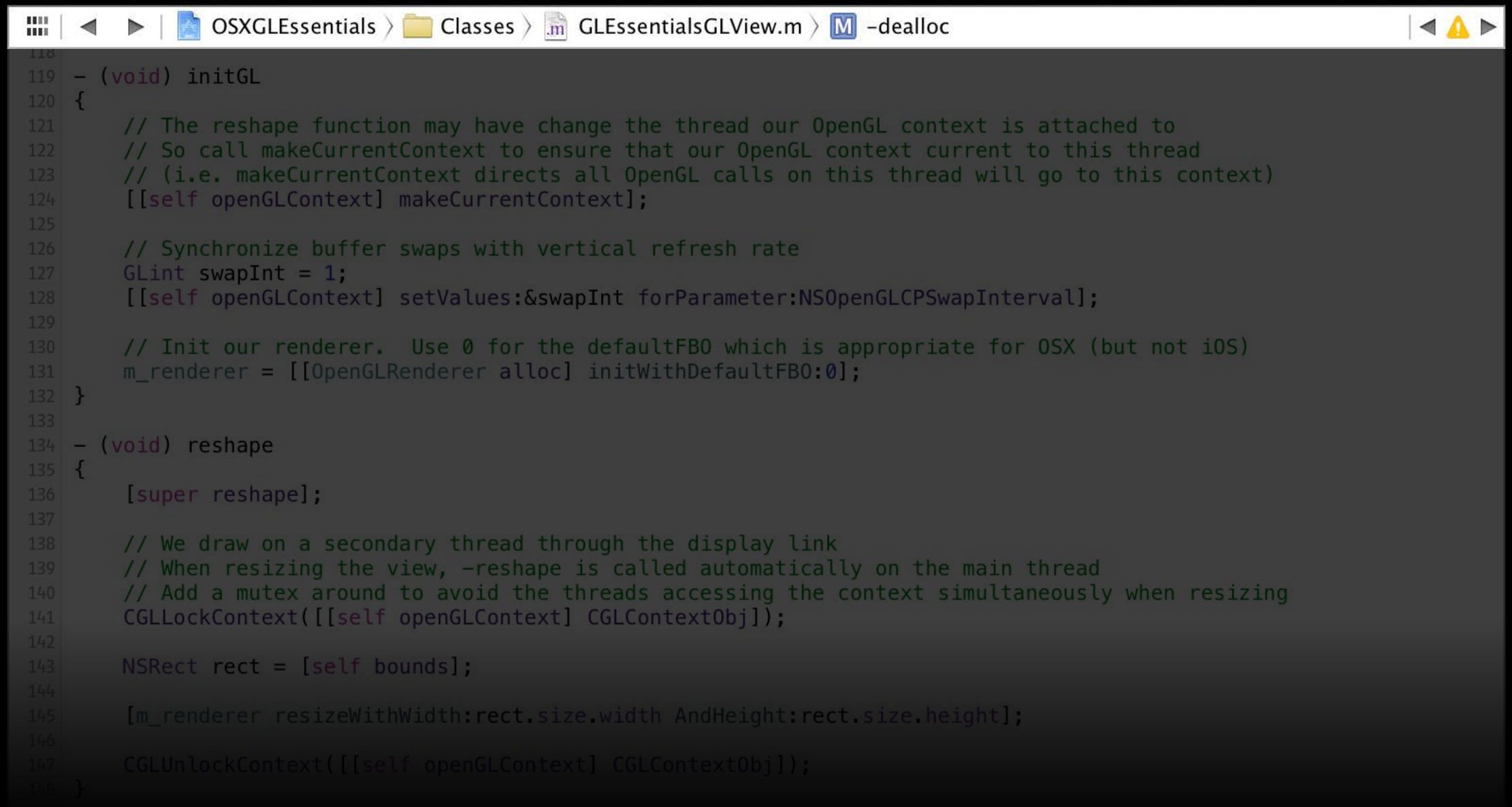



```
119 - (void) initGL
120 {
121     // The reshape function may have change the thread our OpenGL context is attached to
122     // So call makeCurrentContext to ensure that our OpenGL context current to this thread
123     // (i.e. makeCurrentContext directs all OpenGL calls on this thread will go to this context)
124     [[self openGLContext] makeCurrentContext];
125 
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129 
130     // Init our renderer.  Use 0 for the defaultFBO which is appropriate for OSX (but not iOS)
131     m_renderer = [[OpenGLRenderer alloc] initWithDefaultFBO:0];
132 }
133 
134 - (void) reshape
135 {
136     [super reshape];
137 
138     // We draw on a secondary thread through the display link
139     // When resizing the view, -reshape is called automatically on the main thread
140     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
141     CGLLockContext([[self openGLContext] CGLContextObj]);
142 
143     NSRect rect = [self bounds];
144 
145     [m_renderer resizeWithWidth:rect.size.width AndHeight:rect.size.height];
146 
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
```

Editor Jump Bar

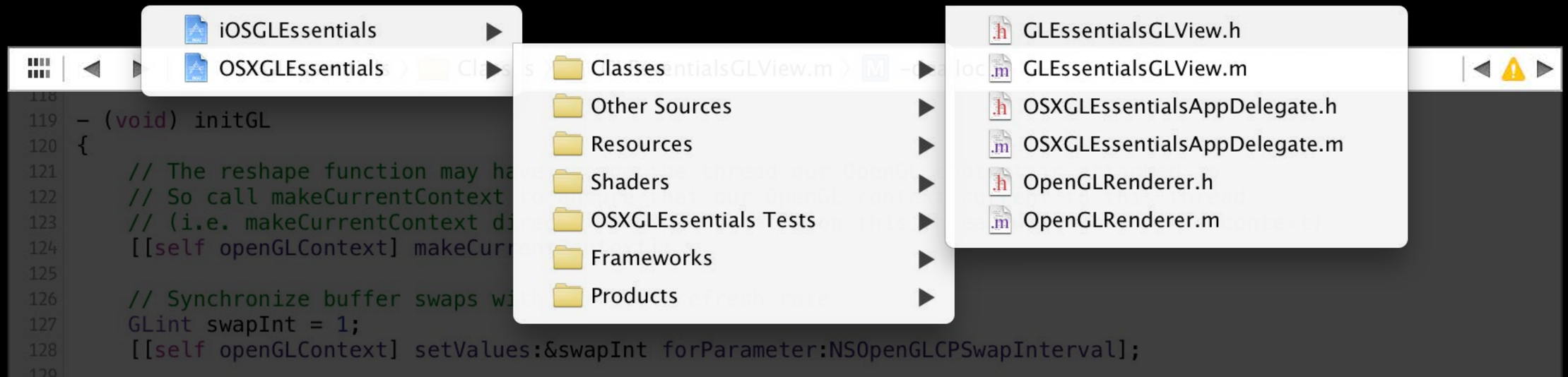


Editor Jump Bar



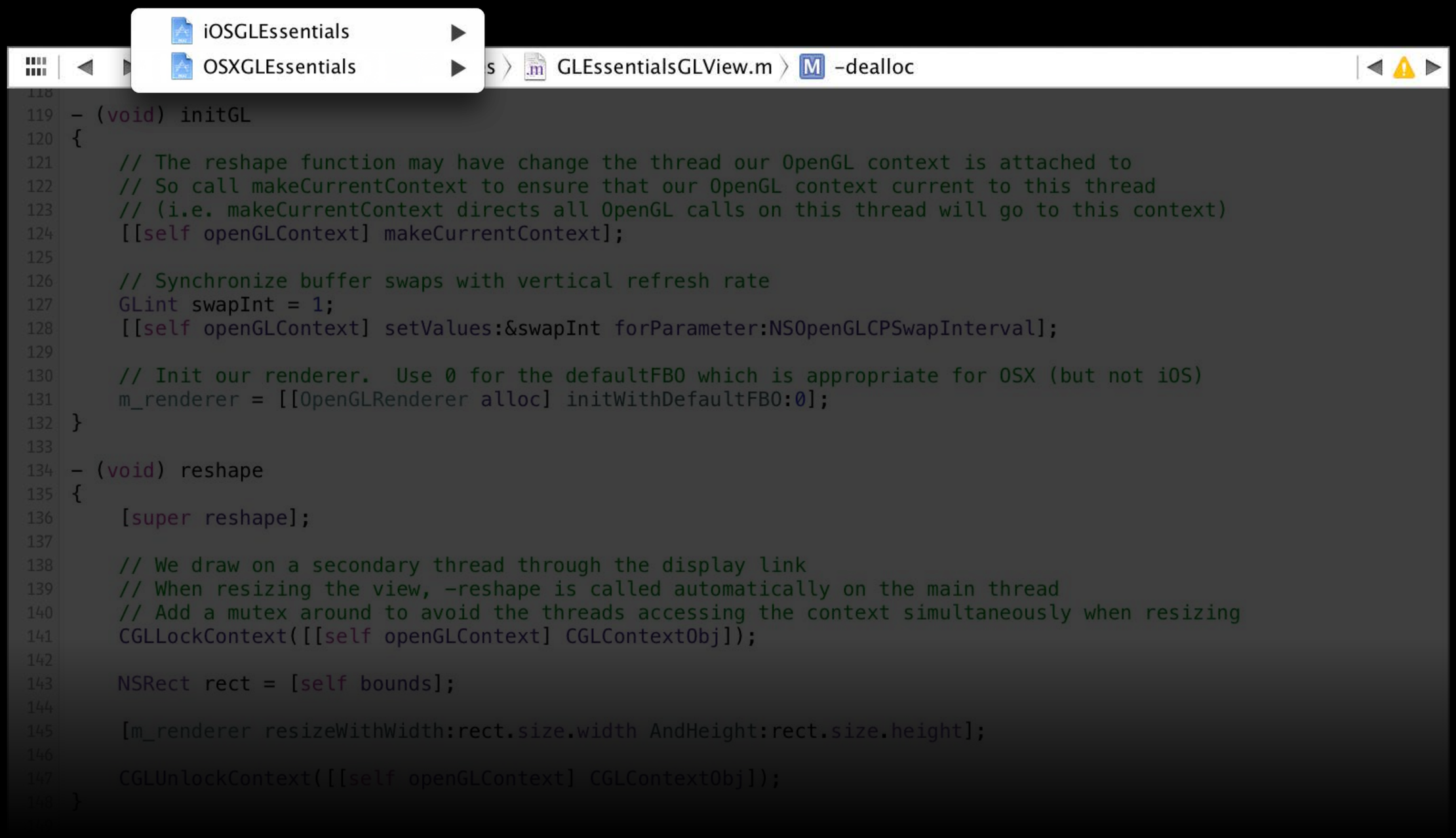

```
119 - (void) initGL
120 {
121     // The reshape function may have change the thread our OpenGL context is attached to
122     // So call makeCurrentContext to ensure that our OpenGL context current to this thread
123     // (i.e. makeCurrentContext directs all OpenGL calls on this thread will go to this context)
124     [[self openGLContext] makeCurrentContext];
125 
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129 
130     // Init our renderer.  Use 0 for the defaultFBO which is appropriate for OSX (but not iOS)
131     m_renderer = [[OpenGLRenderer alloc] initWithDefaultFBO:0];
132 }
133 
134 - (void) reshape
135 {
136     [super reshape];
137 
138     // We draw on a secondary thread through the display link
139     // When resizing the view, -reshape is called automatically on the main thread
140     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
141     CGLLockContext([[self openGLContext] CGLContextObj]);
142 
143     NSRect rect = [self bounds];
144 
145     [m_renderer resizeWithWidth:rect.size.width AndHeight:rect.size.height];
146 
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
```

Editor Jump Bar



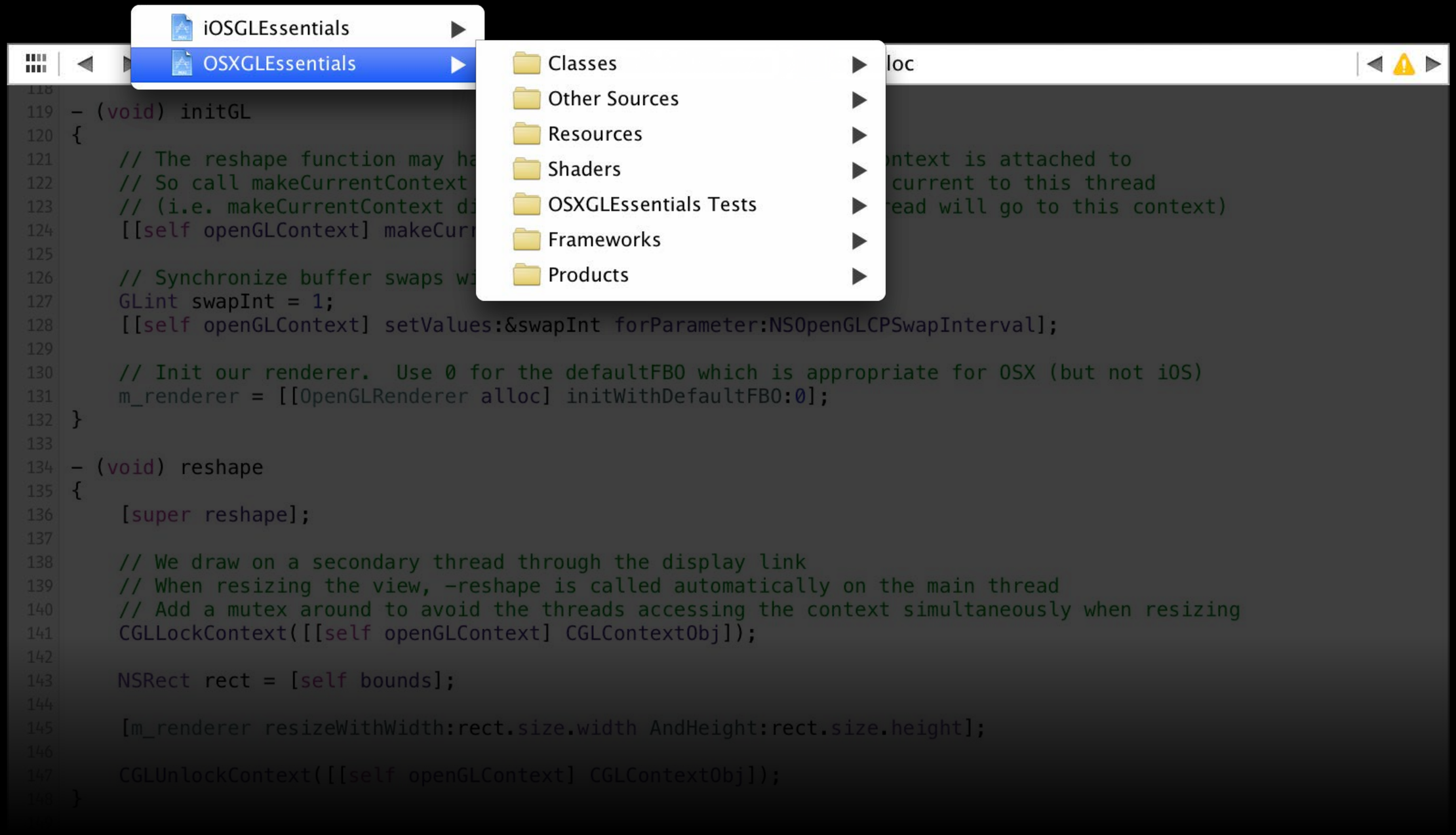

```
- (void) initGL
{
    // The reshape function may ha
    // So call makeCurrentContext
    // (i.e. makeCurrentContext di
    [[self openGLContext] makeCurr

    // Synchronize buffer swaps wi
    GLint swapInt = 1;
    [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];

    // Init our renderer.  Use 0 for the defaultFBO which is appropriate for OSX (but not iOS)
    m_renderer = [[OpenGLRenderer alloc] initWithDefaultFBO:0];
}

- (void) reshape
{
    [super reshape];

    // We draw on a secondary thread through the display link
    // When resizing the view, -reshape is called automatically on the main thread
    // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
    CGLLockContext([[self openGLContext] CGLContextObj]);

    NSRect rect = [self bounds];

    [m_renderer resizeWithWidth:rect.size.width AndHeight:rect.size.height];

    CGLUnlockContext([[self openGLContext] CGLContextObj]);
}
```

Editor Jump Bar

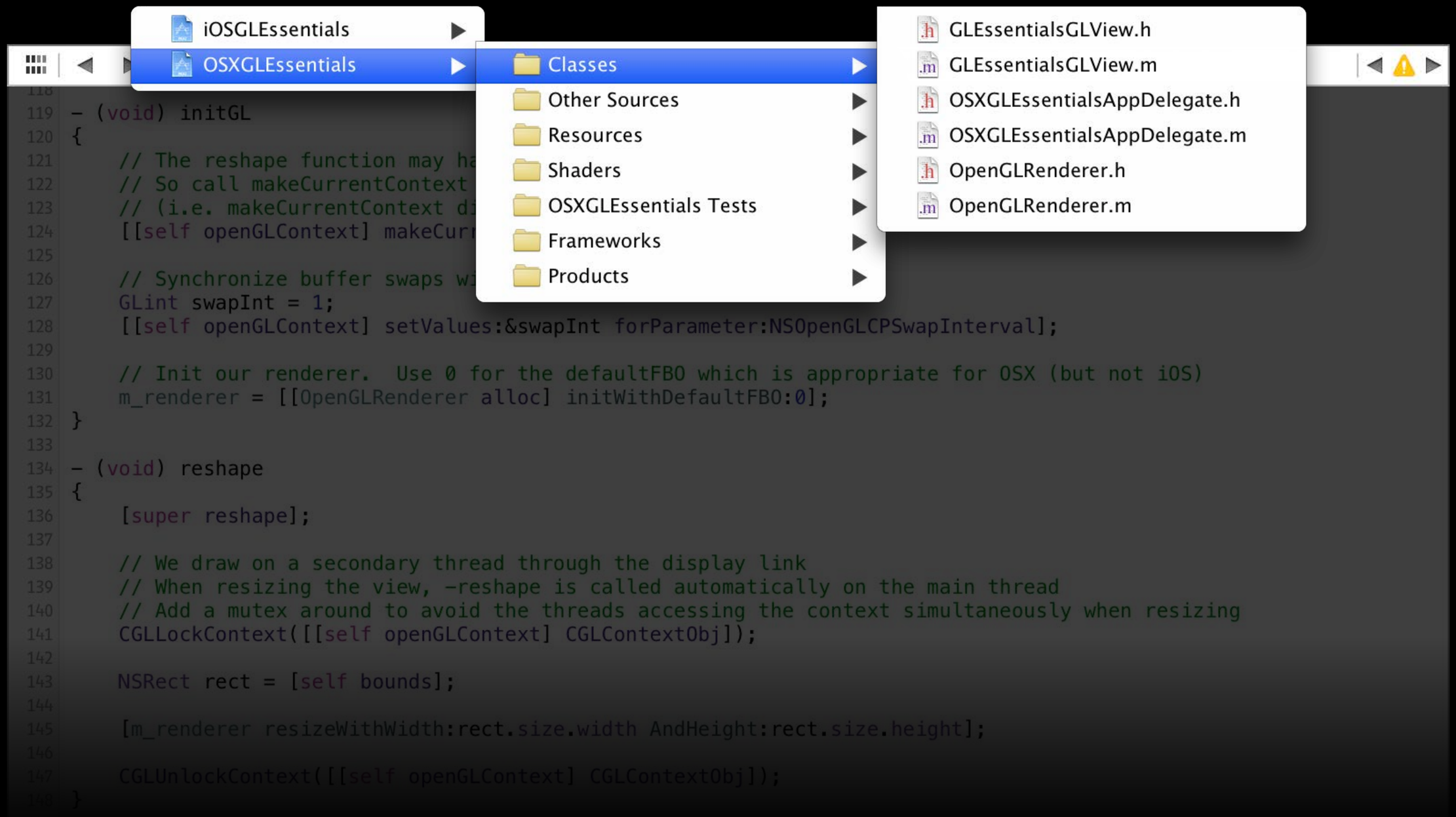



```
- (void) initGL
{
    // The reshape function may have change the thread our OpenGL context is attached to
    // So call makeCurrentContext to ensure that our OpenGL context current to this thread
    // (i.e. makeCurrentContext directs all OpenGL calls on this thread will go to this context)
    [[self openGLContext] makeCurrentContext];

    // Synchronize buffer swaps with vertical refresh rate
    GLint swapInt = 1;
    [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];

    // Init our renderer.  Use 0 for the defaultFBO which is appropriate for OSX (but not iOS)
    m_renderer = [[OpenGLRenderer alloc] initWithDefaultFBO:0];
}

- (void) reshape
{
    [super reshape];

    // We draw on a secondary thread through the display link
    // When resizing the view, -reshape is called automatically on the main thread
    // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
    CGLLockContext([[self openGLContext] CGLContextObj]);

    NSRect rect = [self bounds];

    [m_renderer resizeWithWidth:rect.size.width AndHeight:rect.size.height];

    CGLUnlockContext([[self openGLContext] CGLContextObj]);
}
```

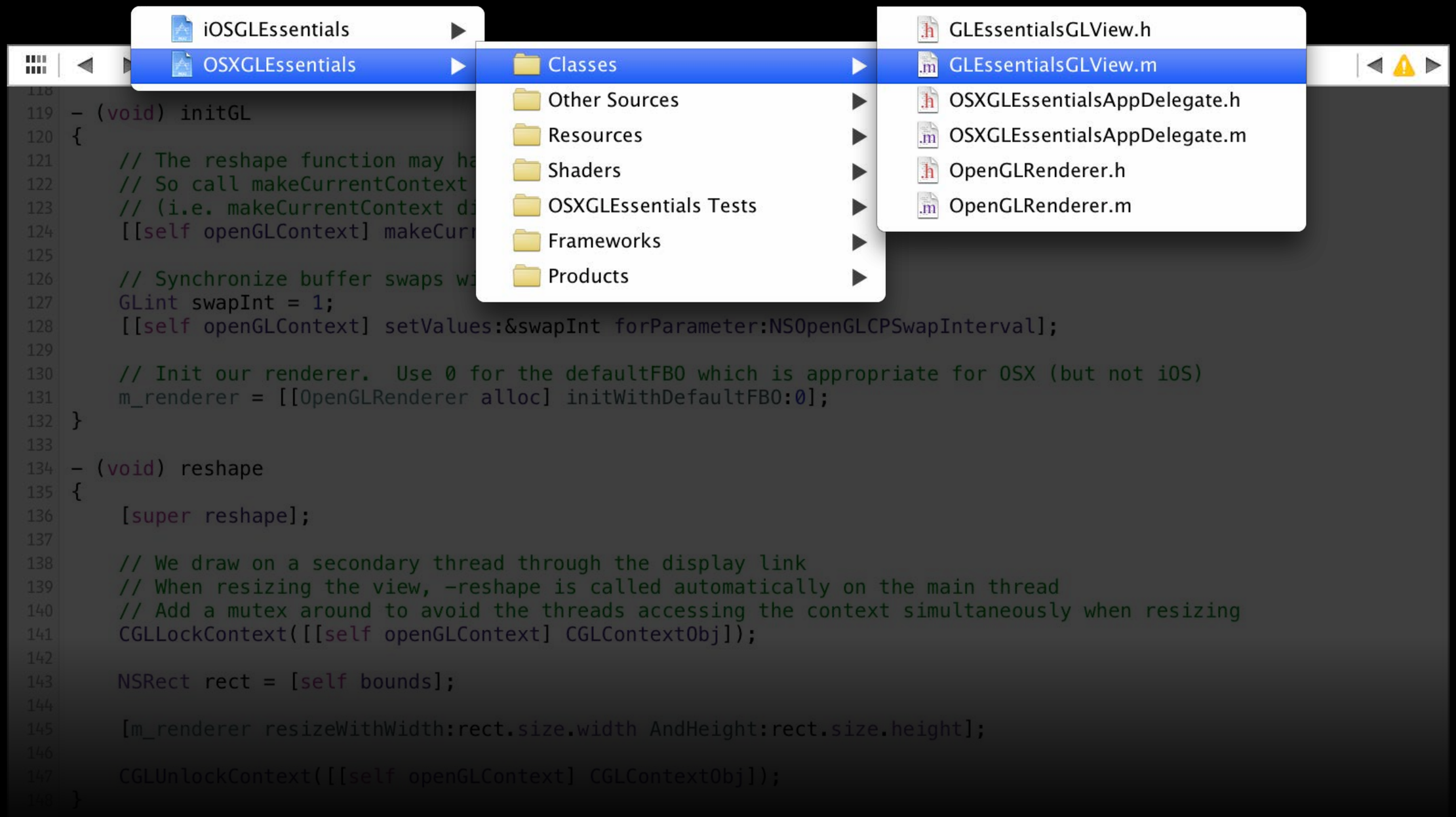
Editor Jump Bar



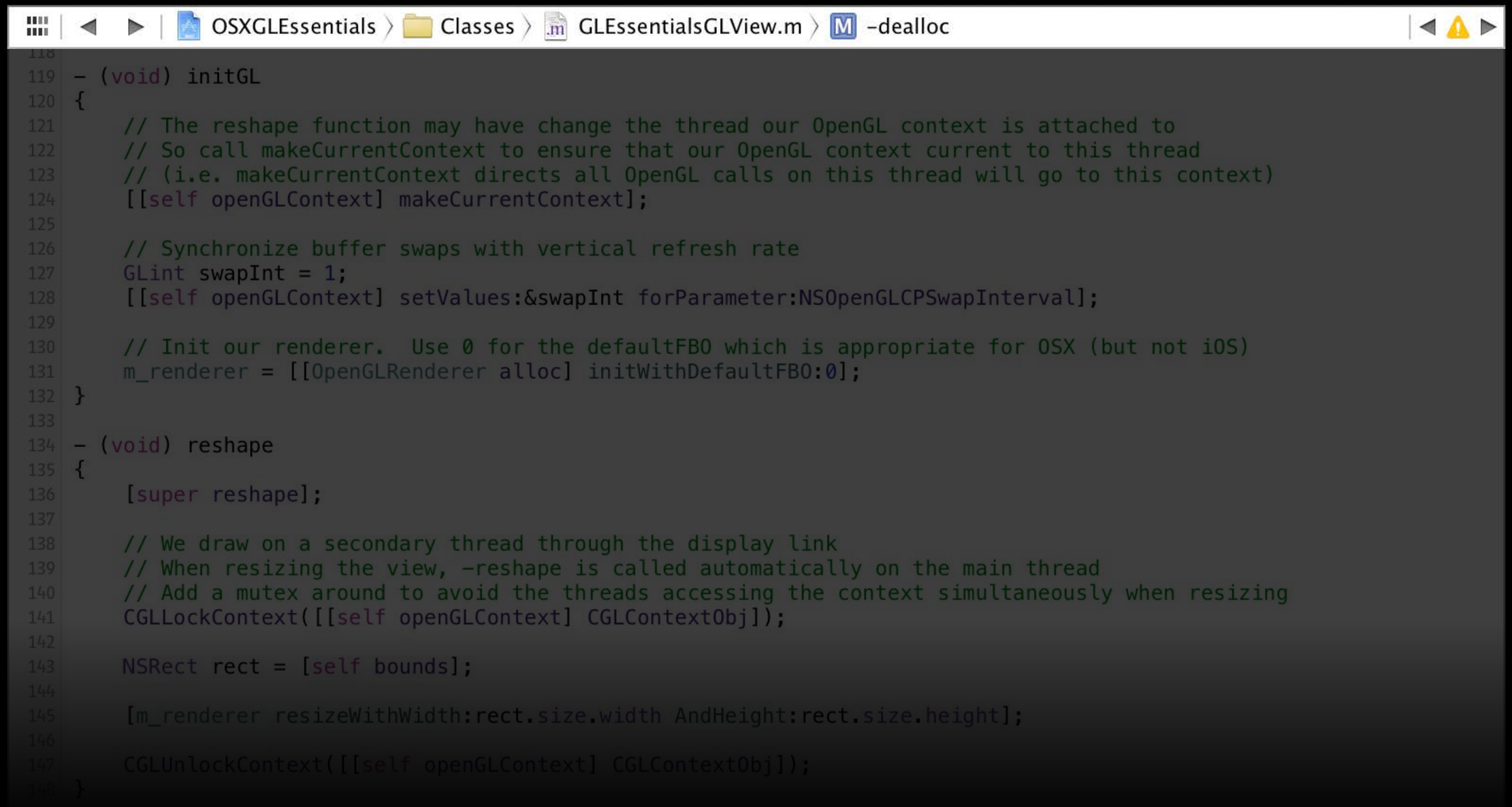
Editor Jump Bar



Editor Jump Bar

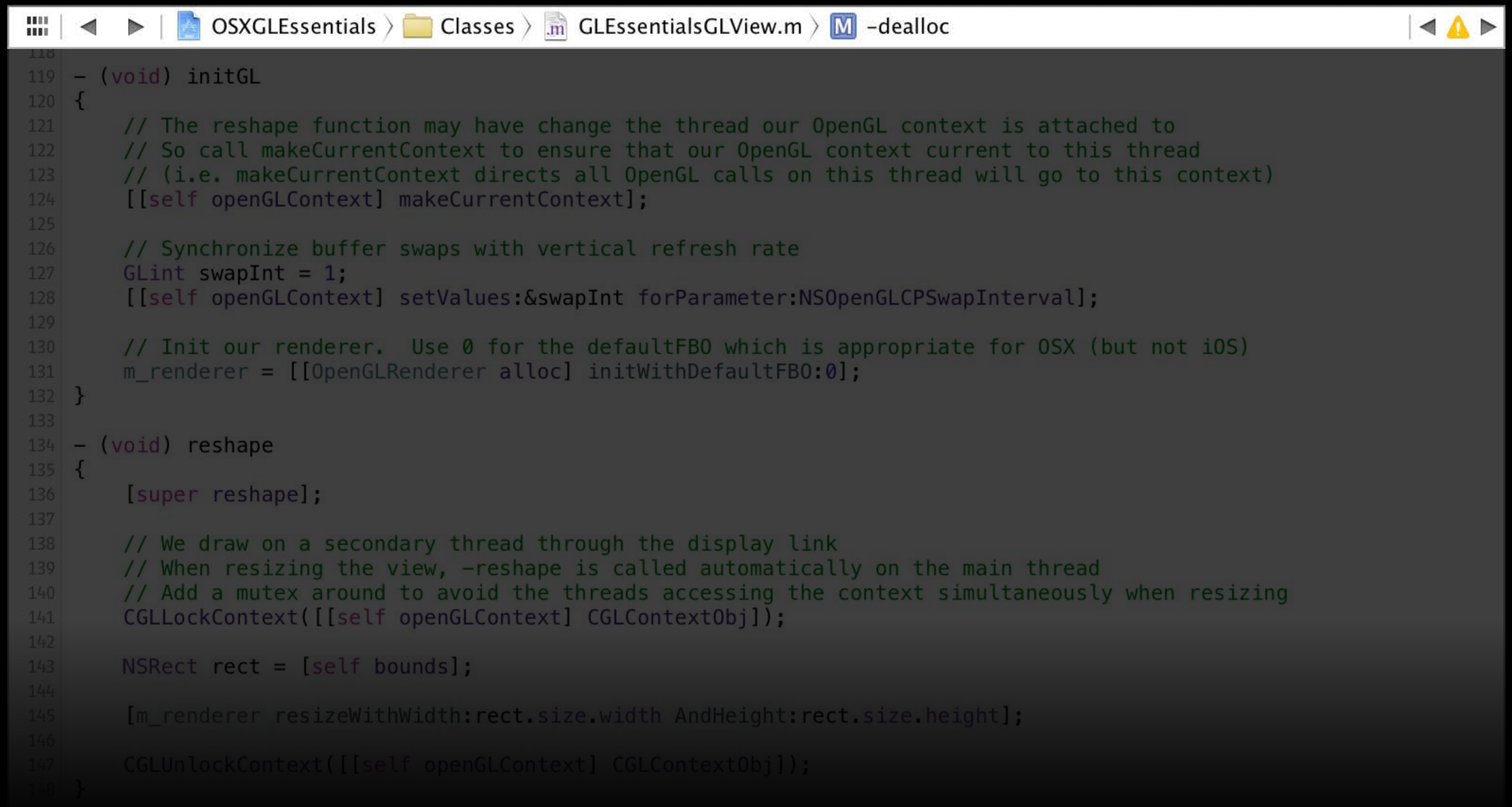


Editor Jump Bar



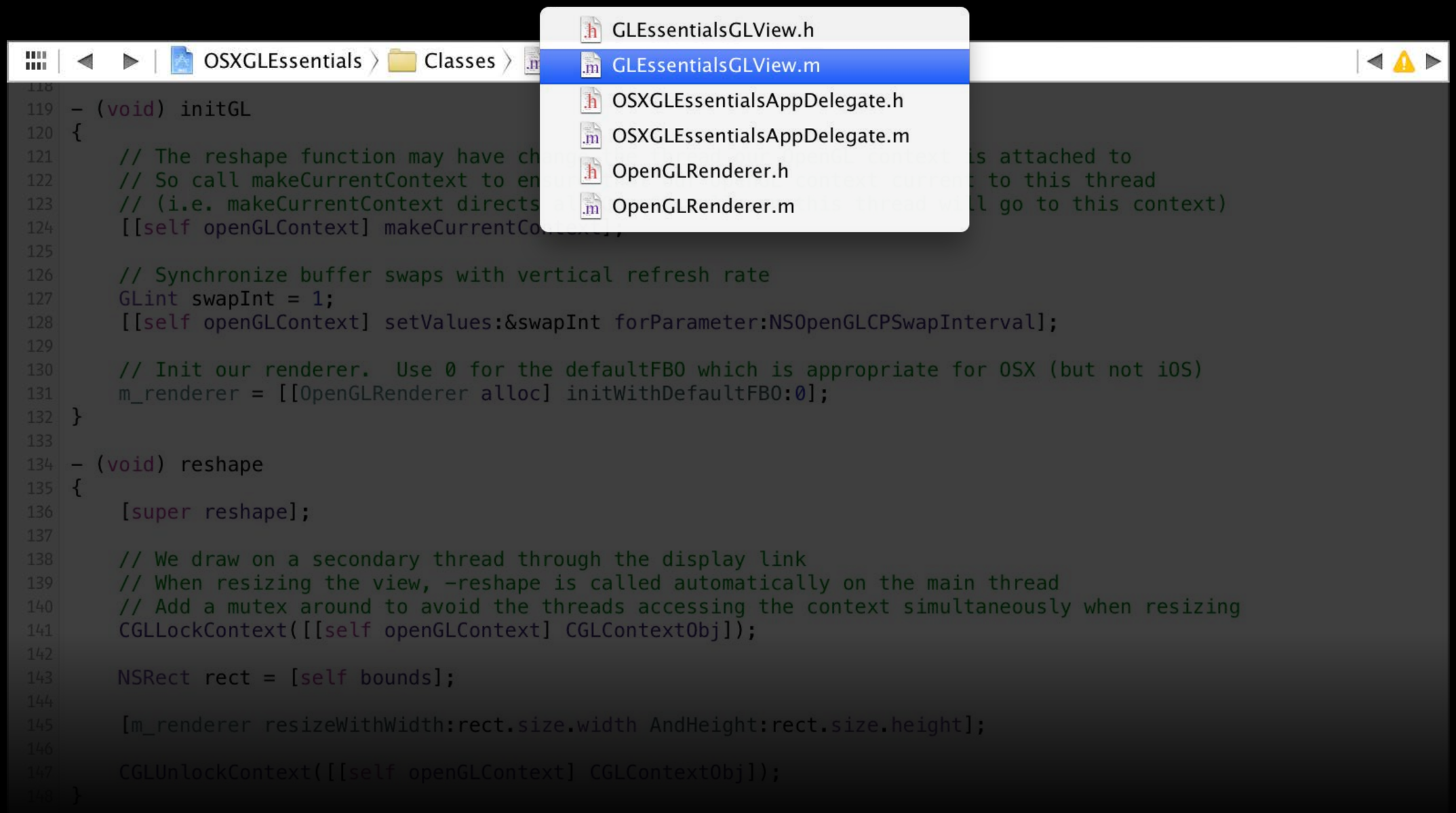

```
119 - (void) initGL
120 {
121     // The reshape function may have change the thread our OpenGL context is attached to
122     // So call makeCurrentContext to ensure that our OpenGL context current to this thread
123     // (i.e. makeCurrentContext directs all OpenGL calls on this thread will go to this context)
124     [[self openGLContext] makeCurrentContext];
125 
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129 
130     // Init our renderer.  Use 0 for the defaultFBO which is appropriate for OSX (but not iOS)
131     m_renderer = [[OpenGLRenderer alloc] initWithDefaultFBO:0];
132 }
133 
134 - (void) reshape
135 {
136     [super reshape];
137 
138     // We draw on a secondary thread through the display link
139     // When resizing the view, -reshape is called automatically on the main thread
140     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
141     CGLLockContext([[self openGLContext] CGLContextObj]);
142 
143     NSRect rect = [self bounds];
144 
145     [m_renderer resizeWithWidth:rect.size.width AndHeight:rect.size.height];
146 
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
```

Editor Jump Bar

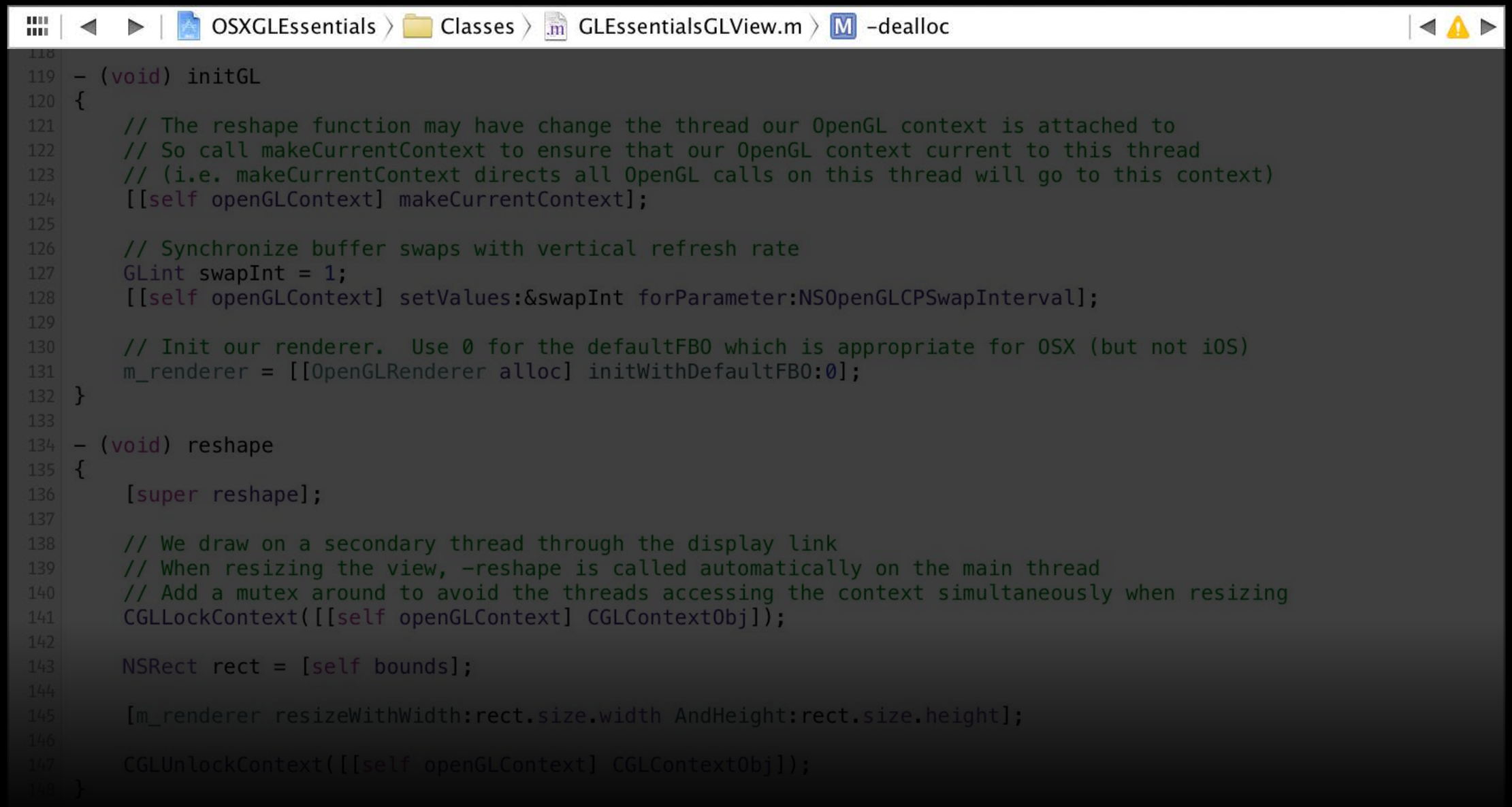



```
119 - (void) initGL
120 {
121     // The reshape function may have change the thread our OpenGL context is attached to
122     // So call makeCurrentContext to ensure that our OpenGL context current to this thread
123     // (i.e. makeCurrentContext directs all OpenGL calls on this thread will go to this context)
124     [[self openGLContext] makeCurrentContext];
125 
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129 
130     // Init our renderer.  Use 0 for the defaultFBO which is appropriate for OSX (but not iOS)
131     m_renderer = [[OpenGLRenderer alloc] initWithDefaultFBO:0];
132 }
133 
134 - (void) reshape
135 {
136     [super reshape];
137 
138     // We draw on a secondary thread through the display link
139     // When resizing the view, -reshape is called automatically on the main thread
140     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
141     CGLLockContext([[self openGLContext] CGLContextObj]);
142 
143     NSRect rect = [self bounds];
144 
145     [m_renderer resizeWithWidth:rect.size.width AndHeight:rect.size.height];
146 
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
```

Editor Jump Bar

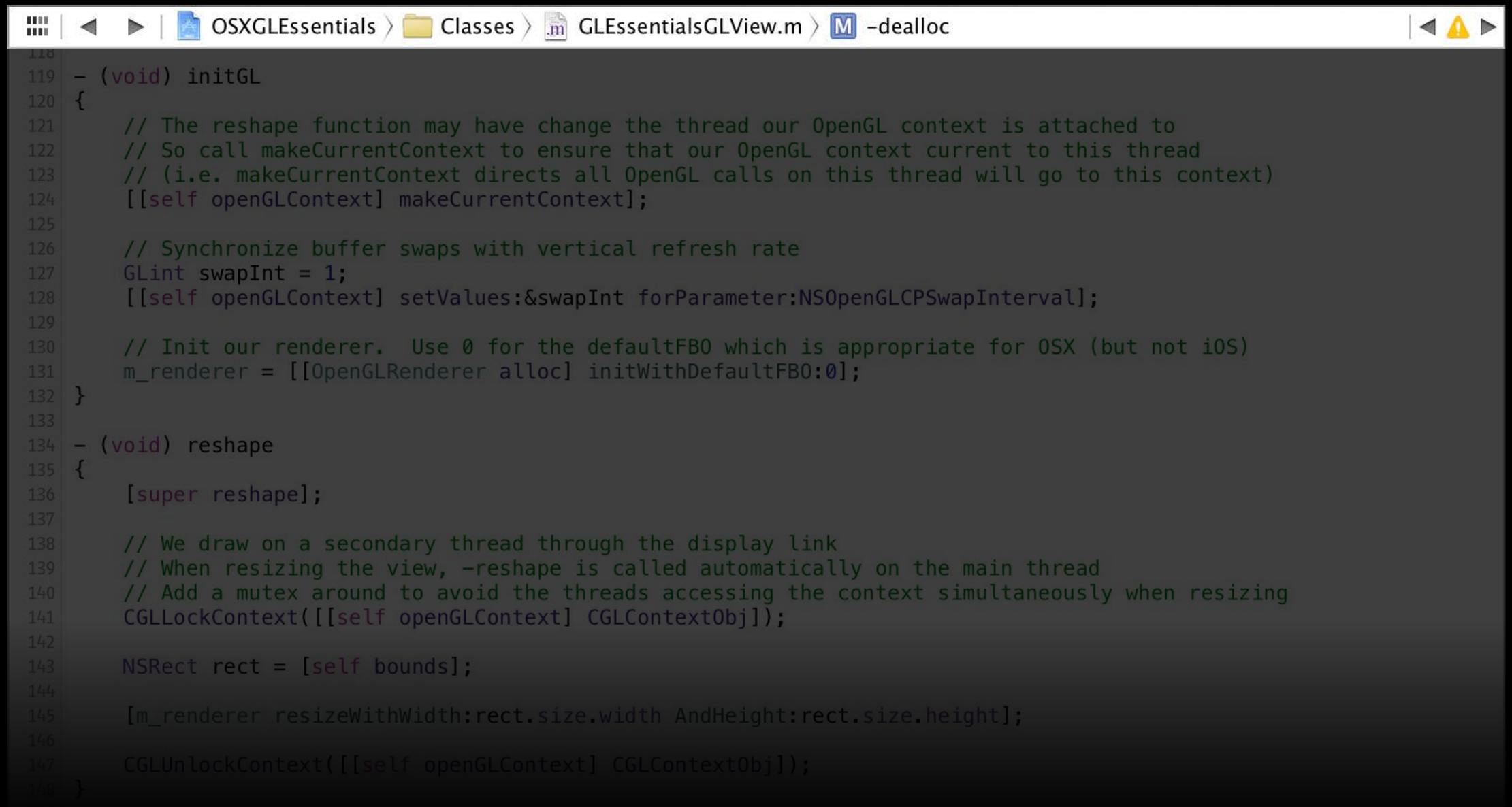


Editor Jump Bar



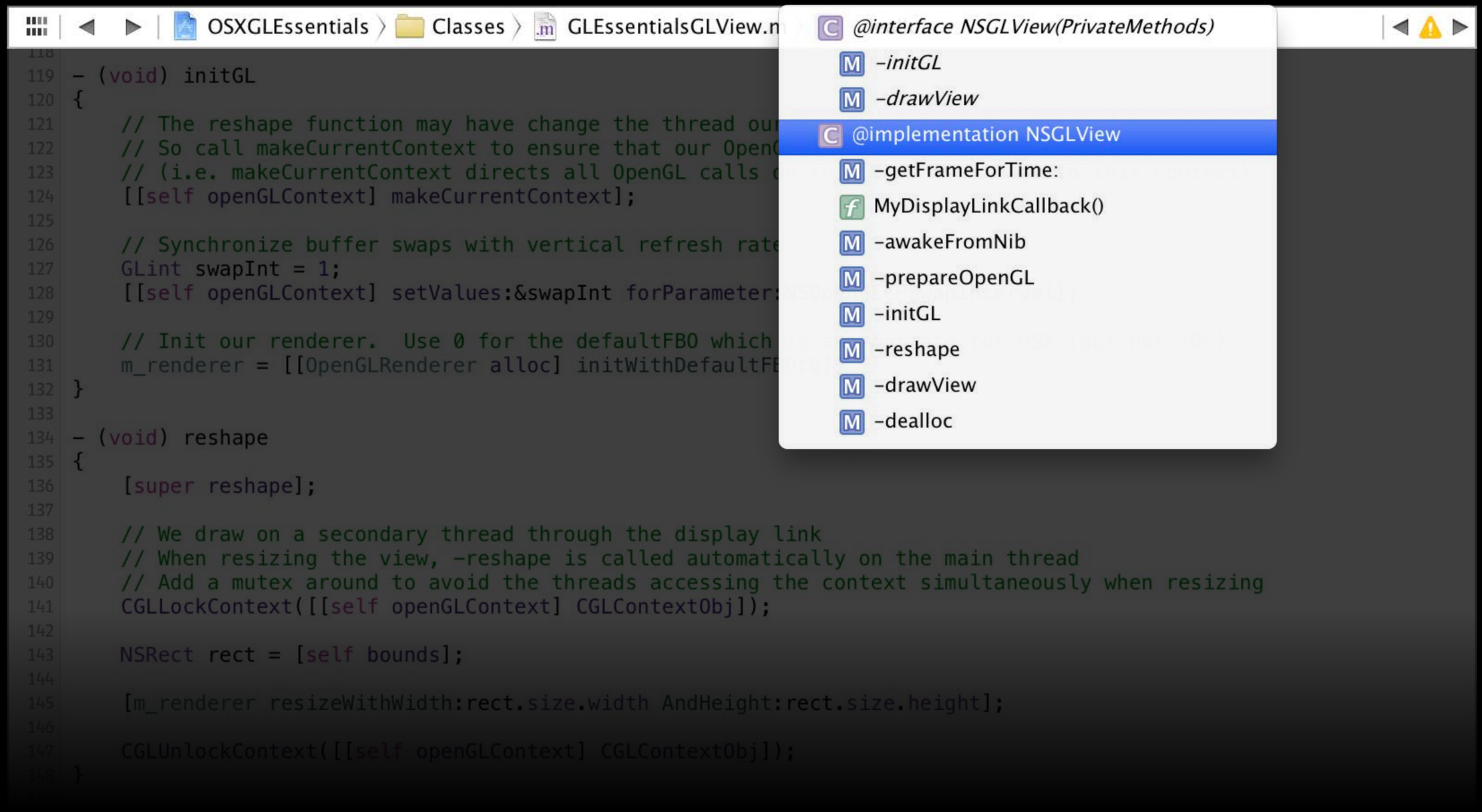

```
119 - (void) initGL
120 {
121     // The reshape function may have change the thread our OpenGL context is attached to
122     // So call makeCurrentContext to ensure that our OpenGL context current to this thread
123     // (i.e. makeCurrentContext directs all OpenGL calls on this thread will go to this context)
124     [[self openGLContext] makeCurrentContext];
125 
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129 
130     // Init our renderer.  Use 0 for the defaultFBO which is appropriate for OSX (but not iOS)
131     m_renderer = [[OpenGLRenderer alloc] initWithDefaultFBO:0];
132 }
133 
134 - (void) reshape
135 {
136     [super reshape];
137 
138     // We draw on a secondary thread through the display link
139     // When resizing the view, -reshape is called automatically on the main thread
140     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
141     CGLLockContext([[self openGLContext] CGLContextObj]);
142 
143     NSRect rect = [self bounds];
144 
145     [m_renderer resizeWithWidth:rect.size.width AndHeight:rect.size.height];
146 
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
```

Editor Jump Bar

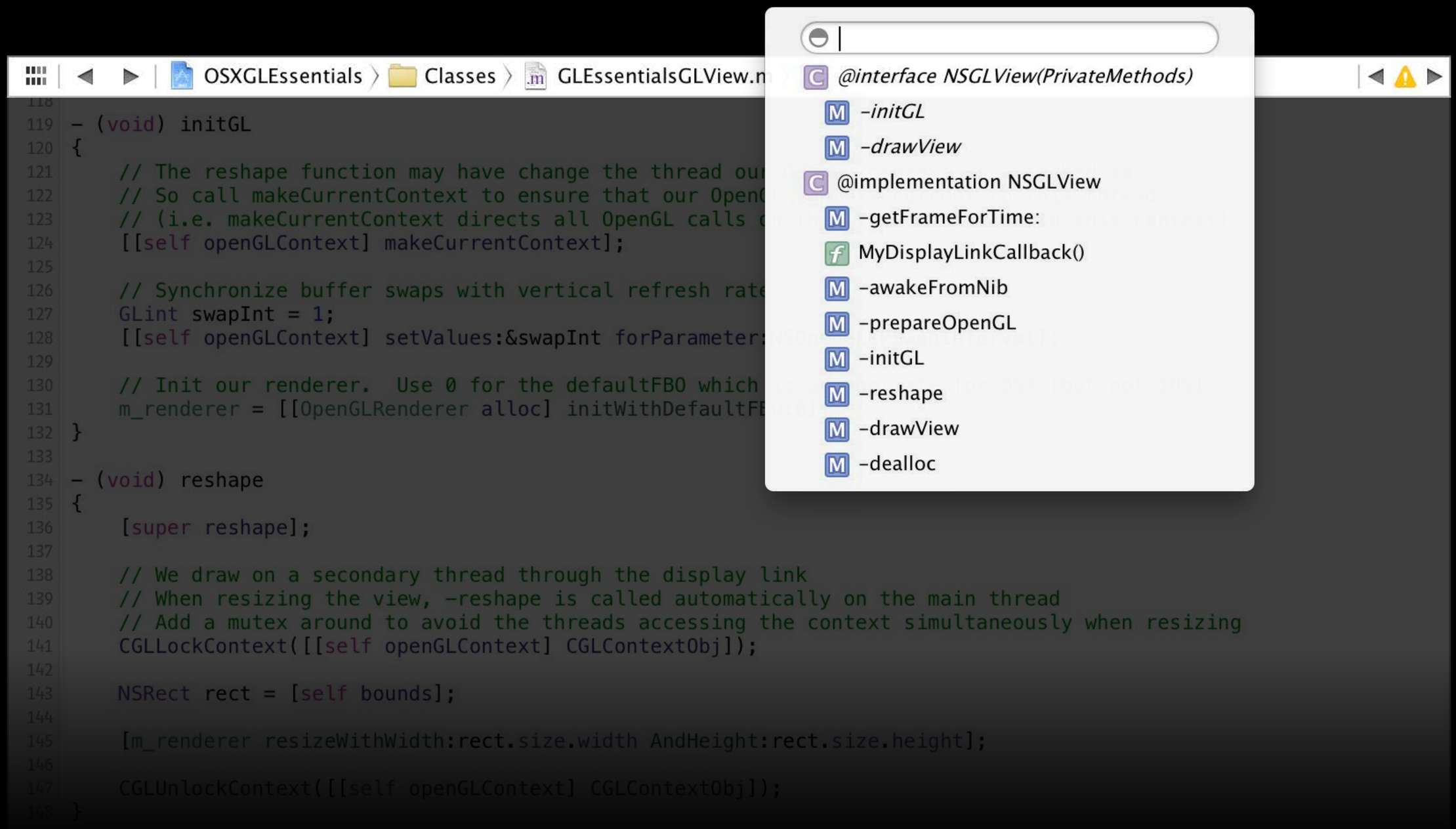



```
119 - (void) initGL
120 {
121     // The reshape function may have change the thread our OpenGL context is attached to
122     // So call makeCurrentContext to ensure that our OpenGL context current to this thread
123     // (i.e. makeCurrentContext directs all OpenGL calls on this thread will go to this context)
124     [[self openGLContext] makeCurrentContext];
125 
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129 
130     // Init our renderer.  Use 0 for the defaultFBO which is appropriate for OSX (but not iOS)
131     m_renderer = [[OpenGLRenderer alloc] initWithDefaultFBO:0];
132 }
133 
134 - (void) reshape
135 {
136     [super reshape];
137 
138     // We draw on a secondary thread through the display link
139     // When resizing the view, -reshape is called automatically on the main thread
140     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
141     CGLLockContext([[self openGLContext] CGLContextObj]);
142 
143     NSRect rect = [self bounds];
144 
145     [m_renderer resizeWithWidth:rect.size.width AndHeight:rect.size.height];
146 
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
```

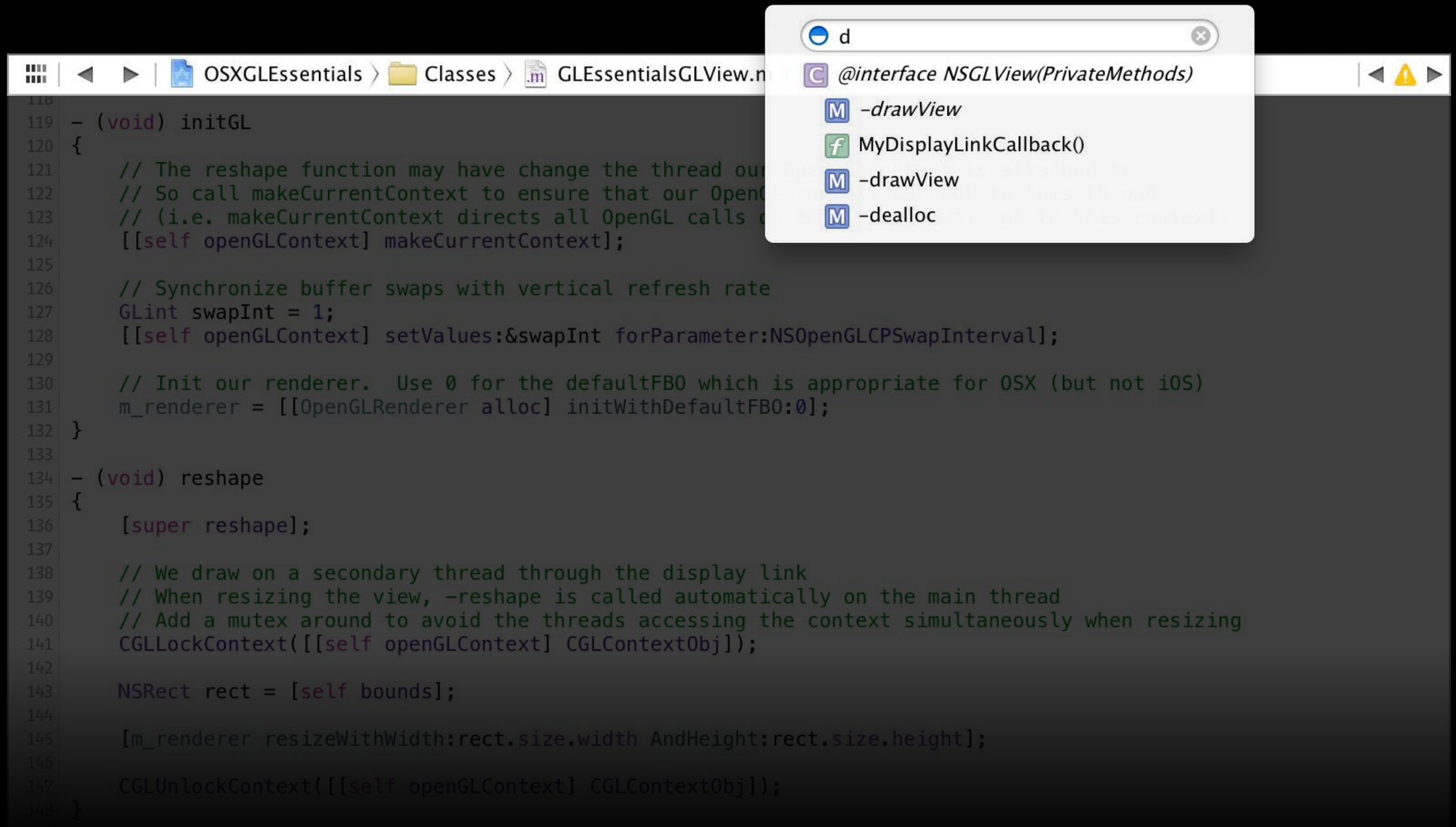
Editor Jump Bar



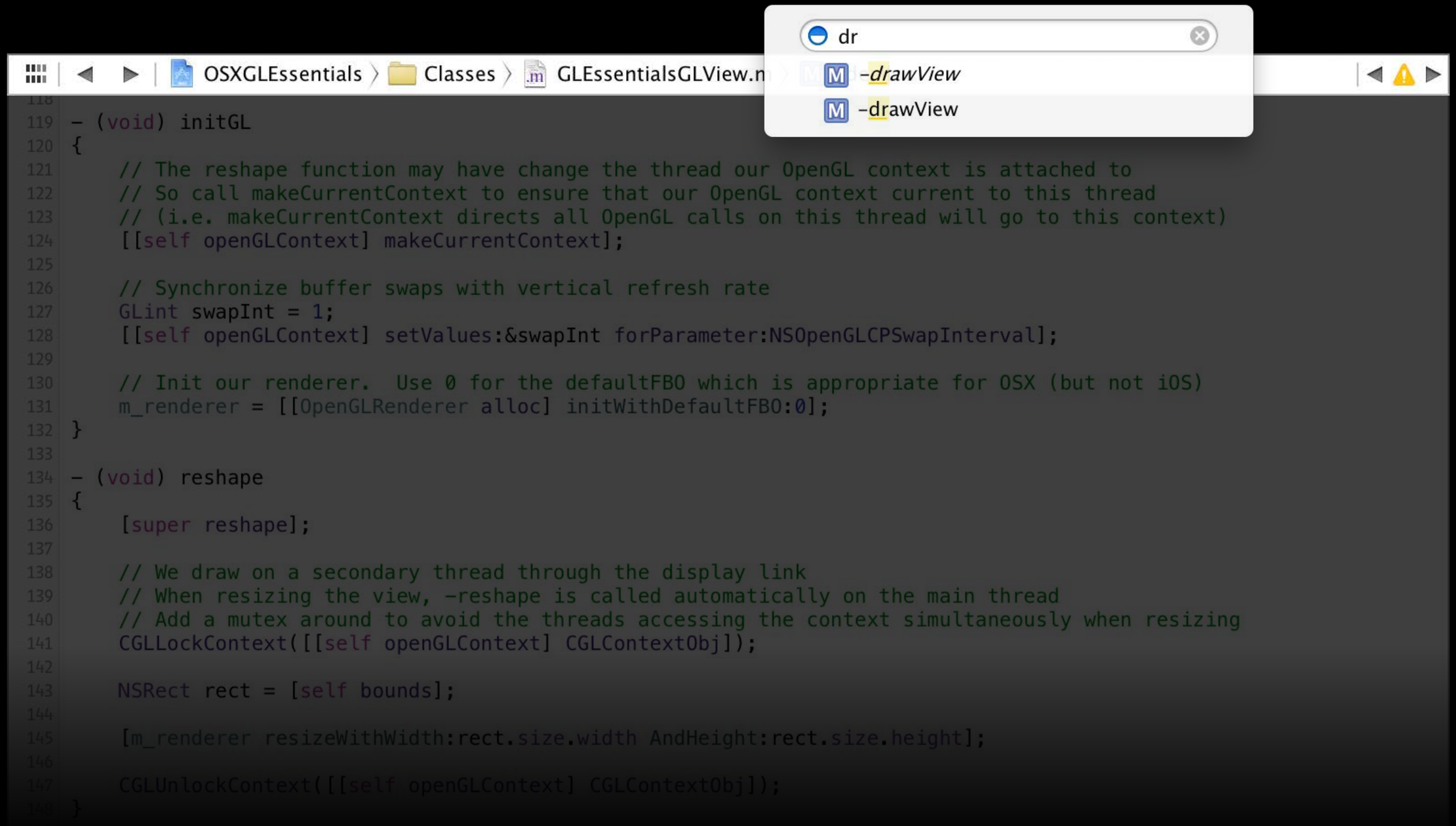
Editor Jump Bar



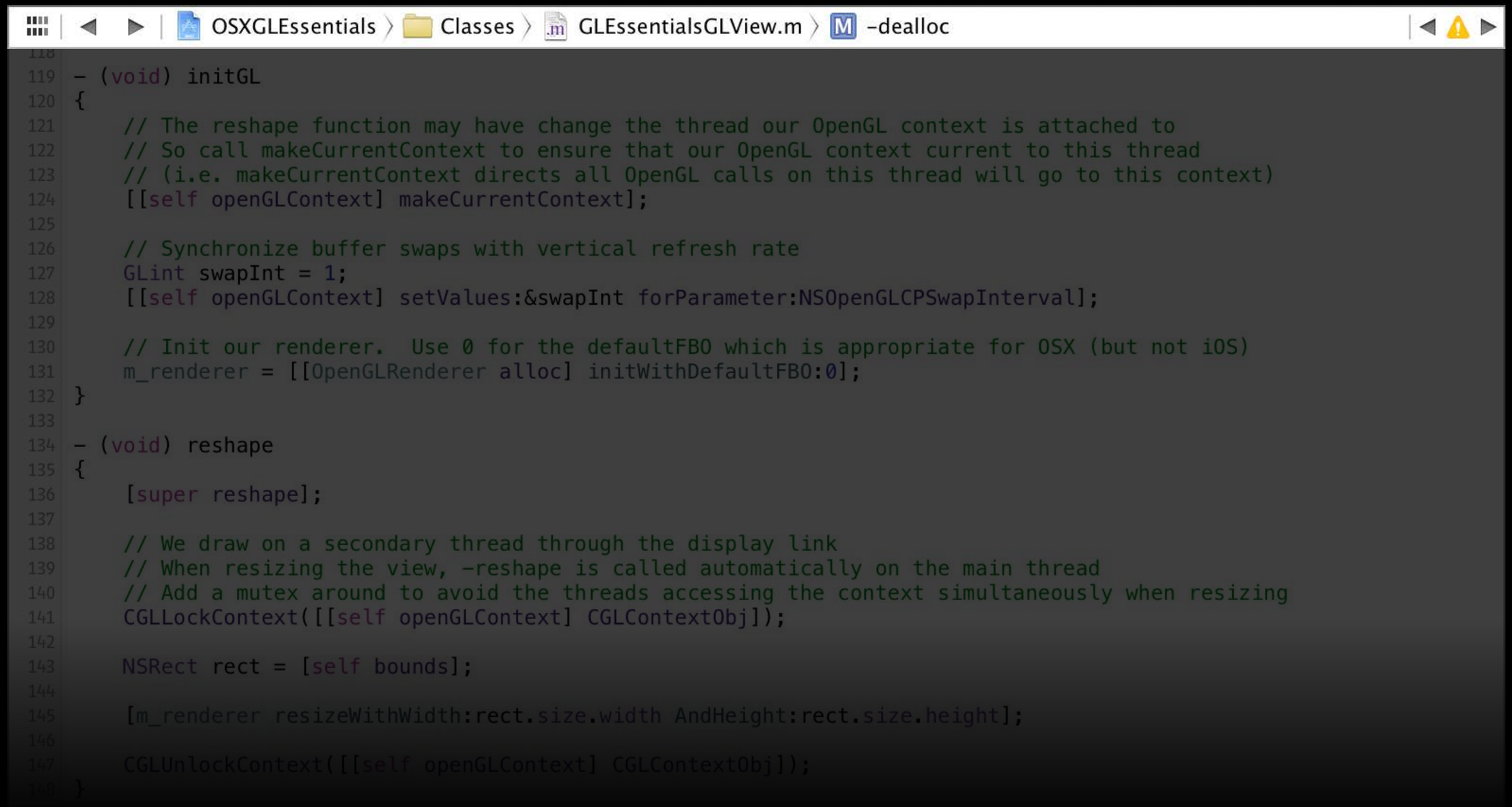
Editor Jump Bar



Editor Jump Bar



Editor Jump Bar




```
119 - (void) initGL
120 {
121     // The reshape function may have change the thread our OpenGL context is attached to
122     // So call makeCurrentContext to ensure that our OpenGL context current to this thread
123     // (i.e. makeCurrentContext directs all OpenGL calls on this thread will go to this context)
124     [[self openGLContext] makeCurrentContext];
125 
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129 
130     // Init our renderer.  Use 0 for the defaultFBO which is appropriate for OSX (but not iOS)
131     m_renderer = [[OpenGLRenderer alloc] initWithDefaultFBO:0];
132 }
133 
134 - (void) reshape
135 {
136     [super reshape];
137 
138     // We draw on a secondary thread through the display link
139     // When resizing the view, -reshape is called automatically on the main thread
140     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
141     CGLLockContext([[self openGLContext] CGLContextObj]);
142 
143     NSRect rect = [self bounds];
144 
145     [m_renderer resizeWithWidth:rect.size.width AndHeight:rect.size.height];
146 
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
```

Assistant Editor

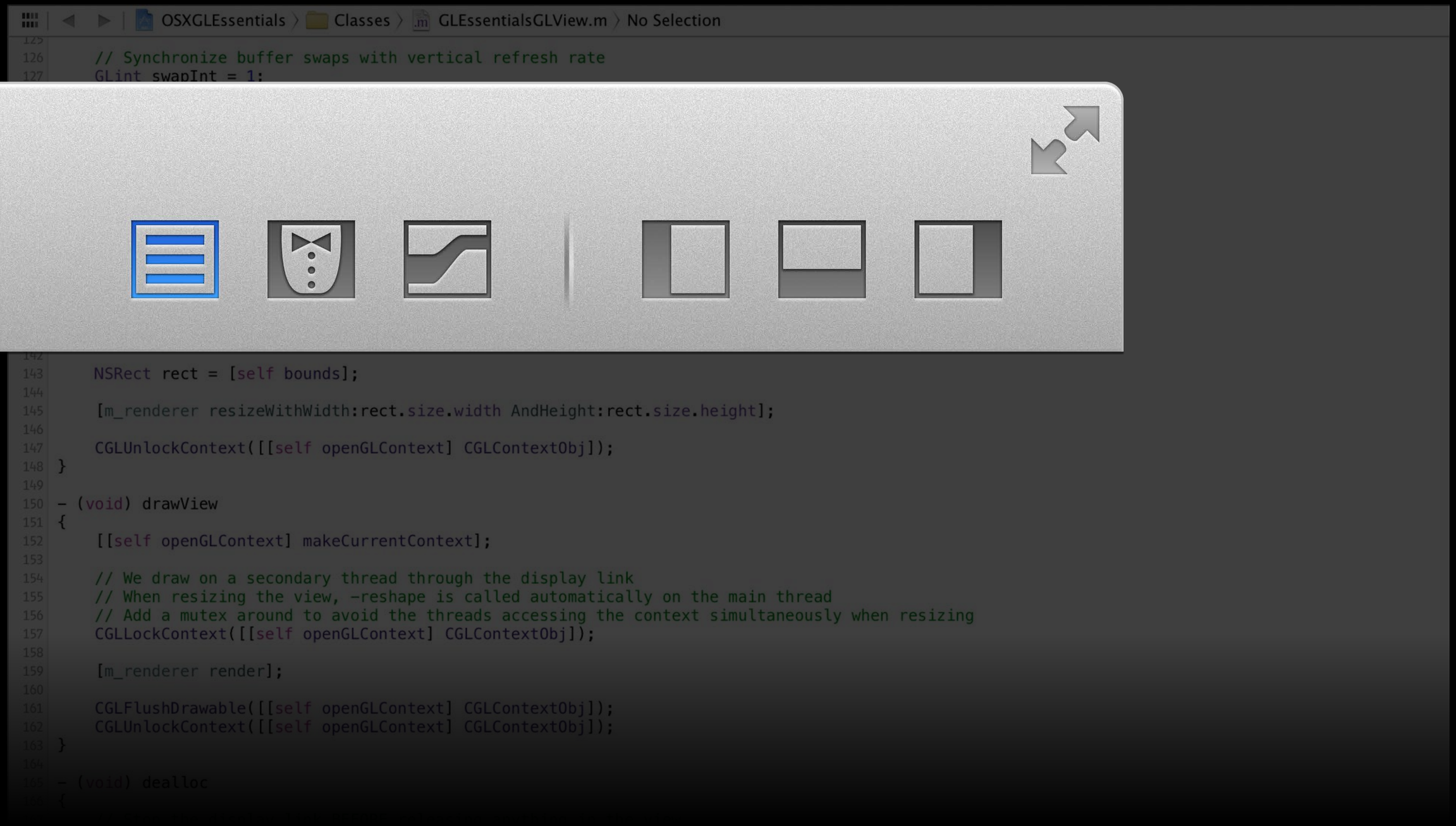
- Secondary (or tertiary) editor
- Tracks contents of main editor
- Flexible editor orientation
- Excellent for auxiliary content

Assistant Editor

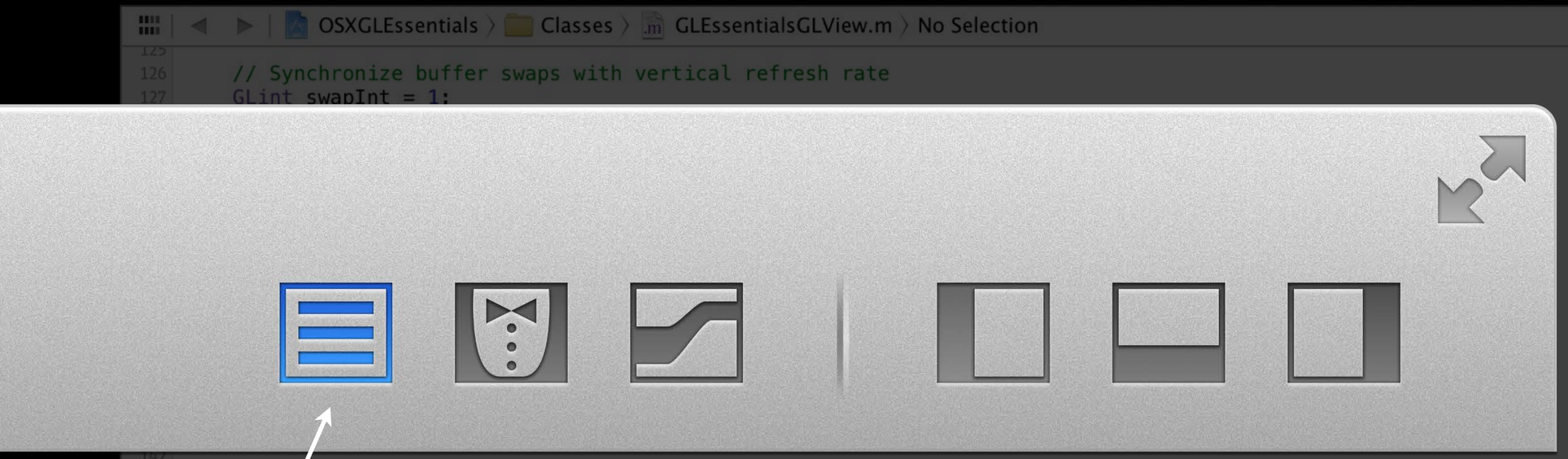
OSXGLEssentials 〉 Classes 〉 GLEssentialsGLView.m 〉 No Selection

```
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129
130     // Init our renderer.  Use 0 for the defaultFBO which is appropriate for OSX (but not iOS)
131     m_renderer = [[OpenGLRenderer alloc] initWithDefaultFBO:0];
132 }
133
134 - (void) reshape
135 {
136     [super reshape];
137
138     // We draw on a secondary thread through the display link
139     // When resizing the view, -reshape is called automatically on the main thread
140     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
141     CGLLockContext([[self openGLContext] CGLContextObj]);
142
143     NSRect rect = [self bounds];
144
145     [m_renderer resizeWithWidth:rect.size.width AndHeight:rect.size.height];
146
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
149
150 - (void) drawView
151 {
152     [[self openGLContext] makeCurrentContext];
153
154     // We draw on a secondary thread through the display link
155     // When resizing the view, -reshape is called automatically on the main thread
156     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
157     CGLLockContext([[self openGLContext] CGLContextObj]);
158
159     [m_renderer render];
160
161     CGLFlushDrawable([[self openGLContext] CGLContextObj]);
162     CGLUnlockContext([[self openGLContext] CGLContextObj]);
163 }
164
165 - (void) dealloc
166 {
```

Assistant Editor



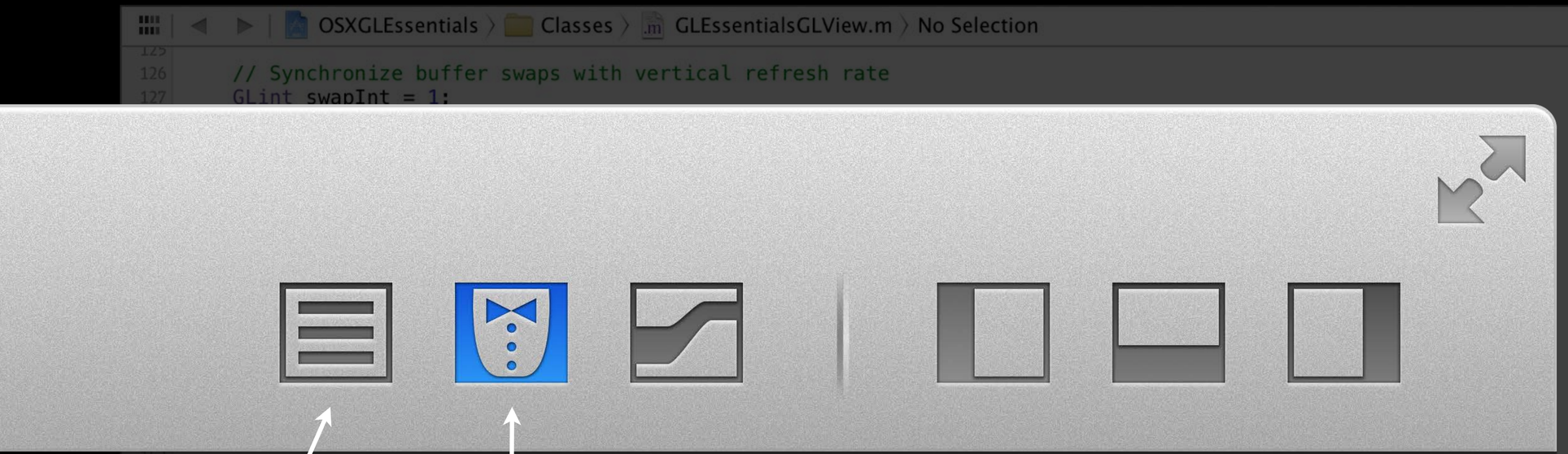
Assistant Editor



Standard Editor

cmd return

Assistant Editor

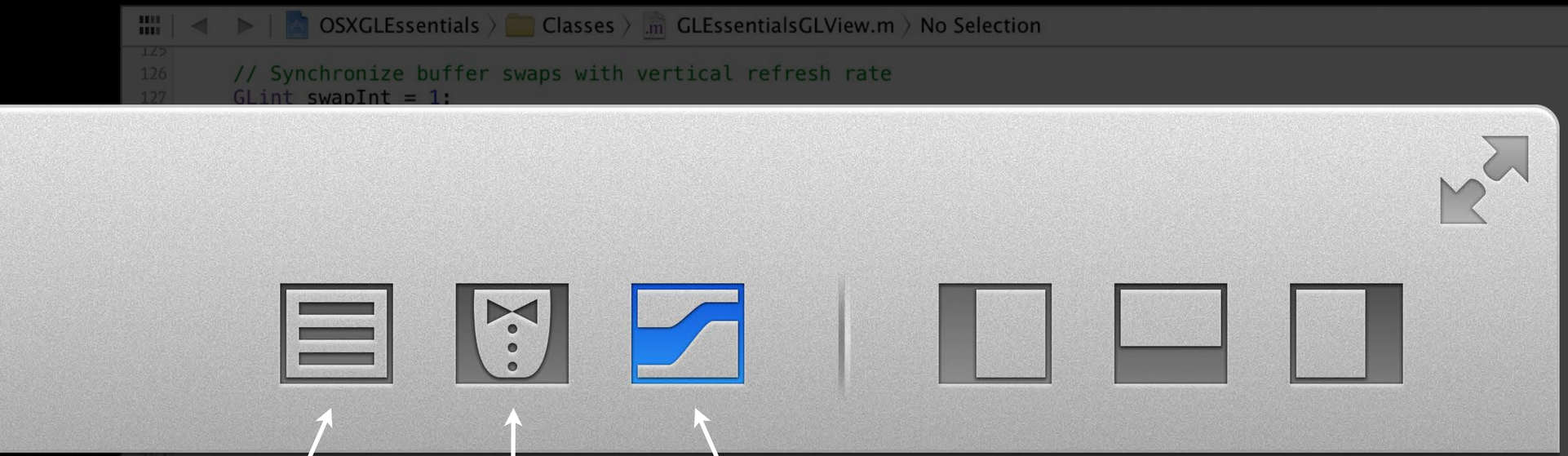


Standard Editor

Assistant Editor

cmd opt return

Assistant Editor



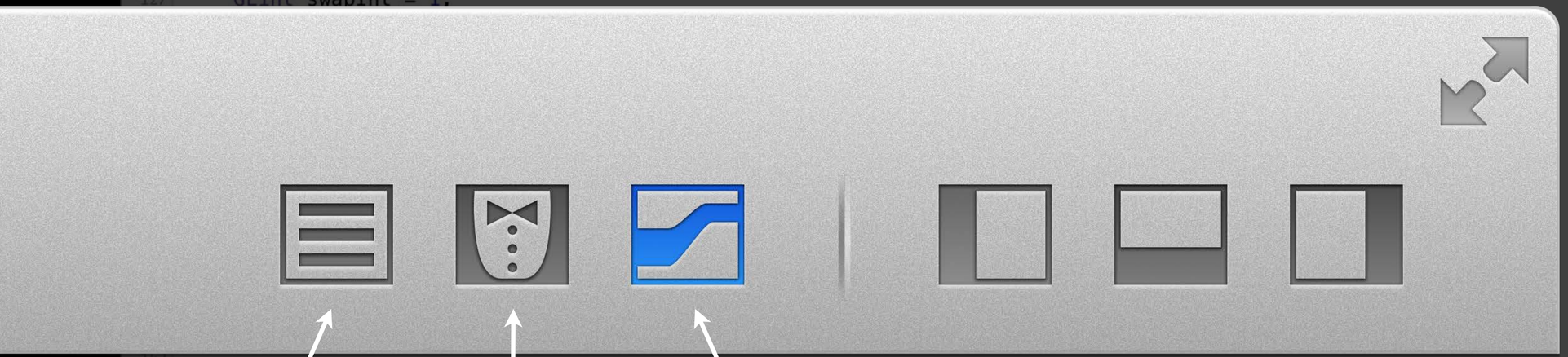
Standard Editor

Assistant Editor

Version Editor

cmd opt shift return

Assistant Editor



Standard Editor

Assistant Editor

Version Editor

Understanding Source Control in Xcode

Pacific Heights
Friday 10:15AM

Assistant Editor

OSXGLEssentials 〉 Classes 〉 GLEssentialsGLView.m 〉 No Selection

```
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129
130     // Init our renderer.  Use 0 for the defaultFBO which is appropriate for OSX (but not iOS)
131     m_renderer = [[OpenGLRenderer alloc] initWithDefaultFBO:0];
132 }
133
134 - (void) reshape
135 {
136     [super reshape];
137
138     // We draw on a secondary thread through the display link
139     // When resizing the view, -reshape is called automatically on the main thread
140     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
141     CGLLockContext([[self openGLContext] CGLContextObj]);
142
143     NSRect rect = [self bounds];
144
145     [m_renderer resizeWithWidth:rect.size.width AndHeight:rect.size.height];
146
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
149
150 - (void) drawView
151 {
152     [[self openGLContext] makeCurrentContext];
153
154     // We draw on a secondary thread through the display link
155     // When resizing the view, -reshape is called automatically on the main thread
156     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
157     CGLLockContext([[self openGLContext] CGLContextObj]);
158
159     [m_renderer render];
160
161     CGLFlushDrawable([[self openGLContext] CGLContextObj]);
162     CGLUnlockContext([[self openGLContext] CGLContextObj]);
163 }
164
165 - (void) dealloc
166 {
```

Assistant Editor

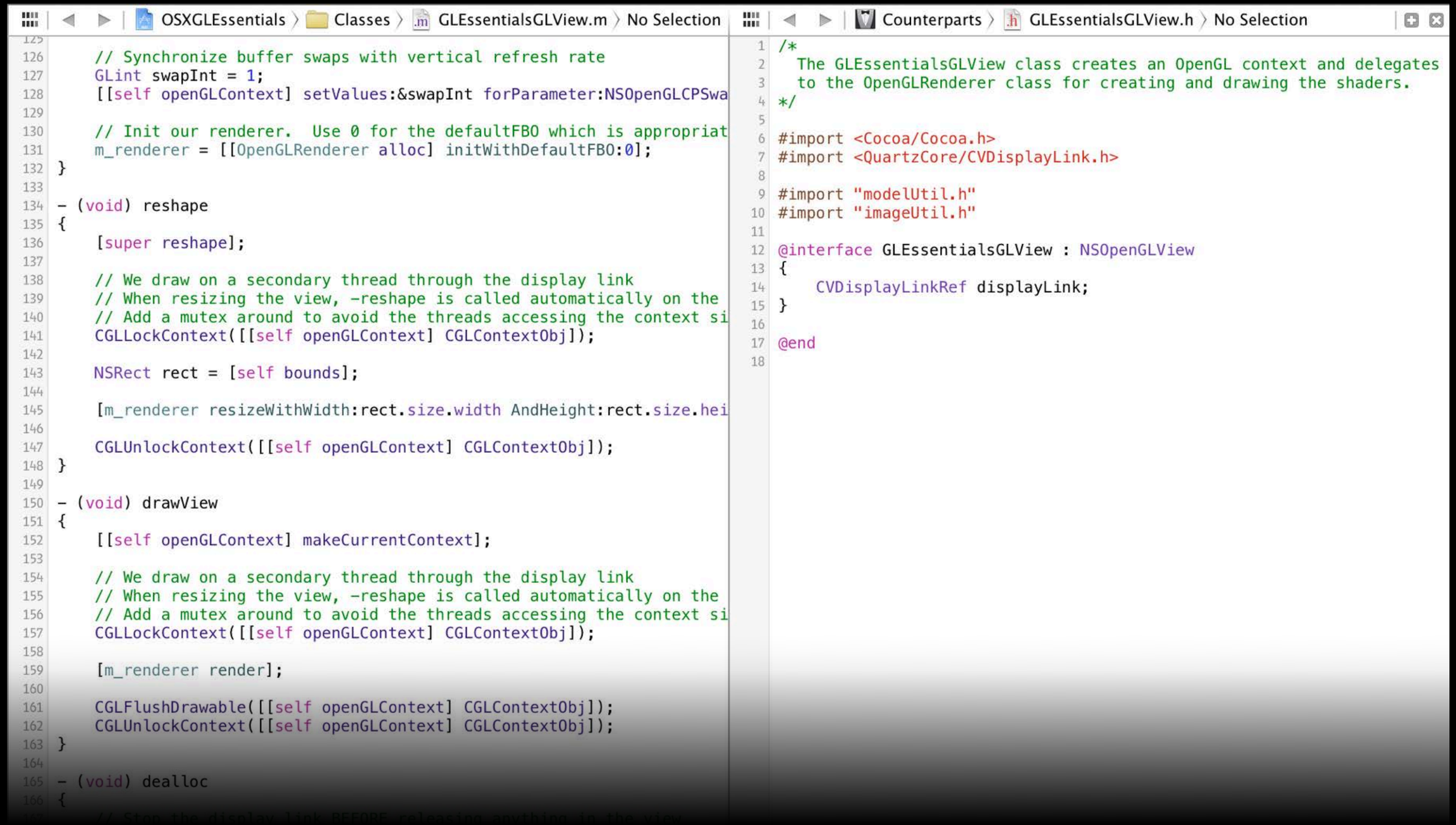
OSXGLEssentials 〉 Classes 〉 GLEssentialsGLView.m 〉 No Selection

```
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwa
129
130     // Init our renderer.  Use 0 for the defaultFBO which is appropriat
131     m_renderer = [[OpenGLRenderer alloc] initWithDefaultFBO:0];
132 }
133
134 - (void) reshape
135 {
136     [super reshape];
137
138     // We draw on a secondary thread through the display link
139     // When resizing the view, -reshape is called automatically on the
140     // Add a mutex around to avoid the threads accessing the context si
141     CGLLockContext([[self openGLContext] CGLContextObj]);
142
143     NSRect rect = [self bounds];
144
145     [m_renderer resizeWithWidth:rect.size.width AndHeight:rect.size.hei
146
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
149
150 - (void) drawView
151 {
152     [[self openGLContext] makeCurrentContext];
153
154     // We draw on a secondary thread through the display link
155     // When resizing the view, -reshape is called automatically on the
156     // Add a mutex around to avoid the threads accessing the context si
157     CGLLockContext([[self openGLContext] CGLContextObj]);
158
159     [m_renderer render];
160
161     CGLFlushDrawable([[self openGLContext] CGLContextObj]);
162     CGLUnlockContext([[self openGLContext] CGLContextObj]);
163 }
164
165 - (void) dealloc
166 {
```

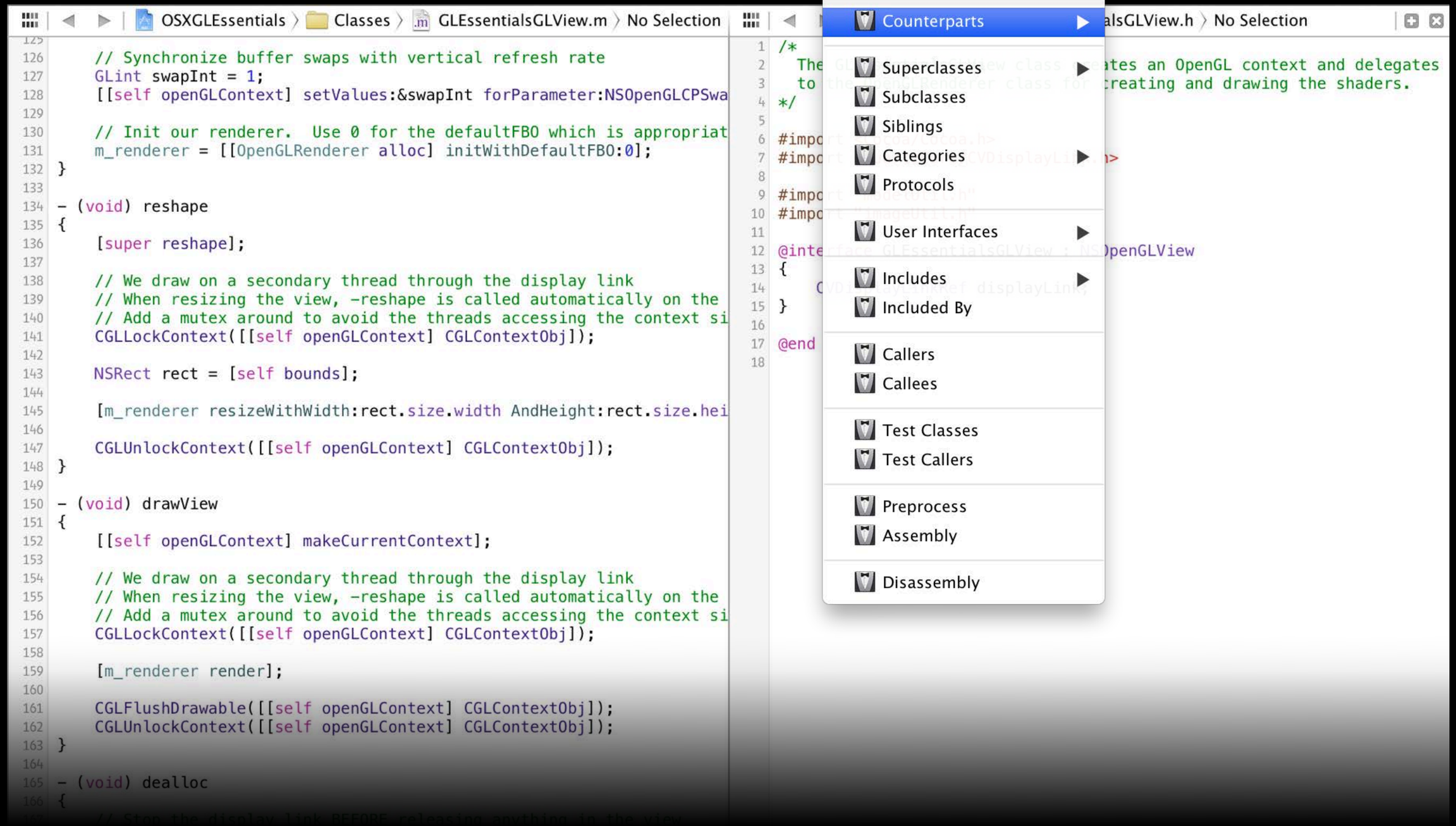
Counterparts 〉 GLEssentialsGLView.h 〉 No Selection

```
1  /*
2    The GLEssentialsGLView class creates an OpenGL context and delegates
3    to the OpenGLRenderer class for creating and drawing the shaders.
4  */
5
6  #import <Cocoa/Cocoa.h>
7  #import <QuartzCore/CVDisplayLink.h>
8
9  #import "modelUtil.h"
10 #import "imageUtil.h"
11
12 @interface GLEssentialsGLView : NSOpenGLView
13 {
14     CVDisplayLinkRef displayLink;
15 }
16
17 @end
18
```

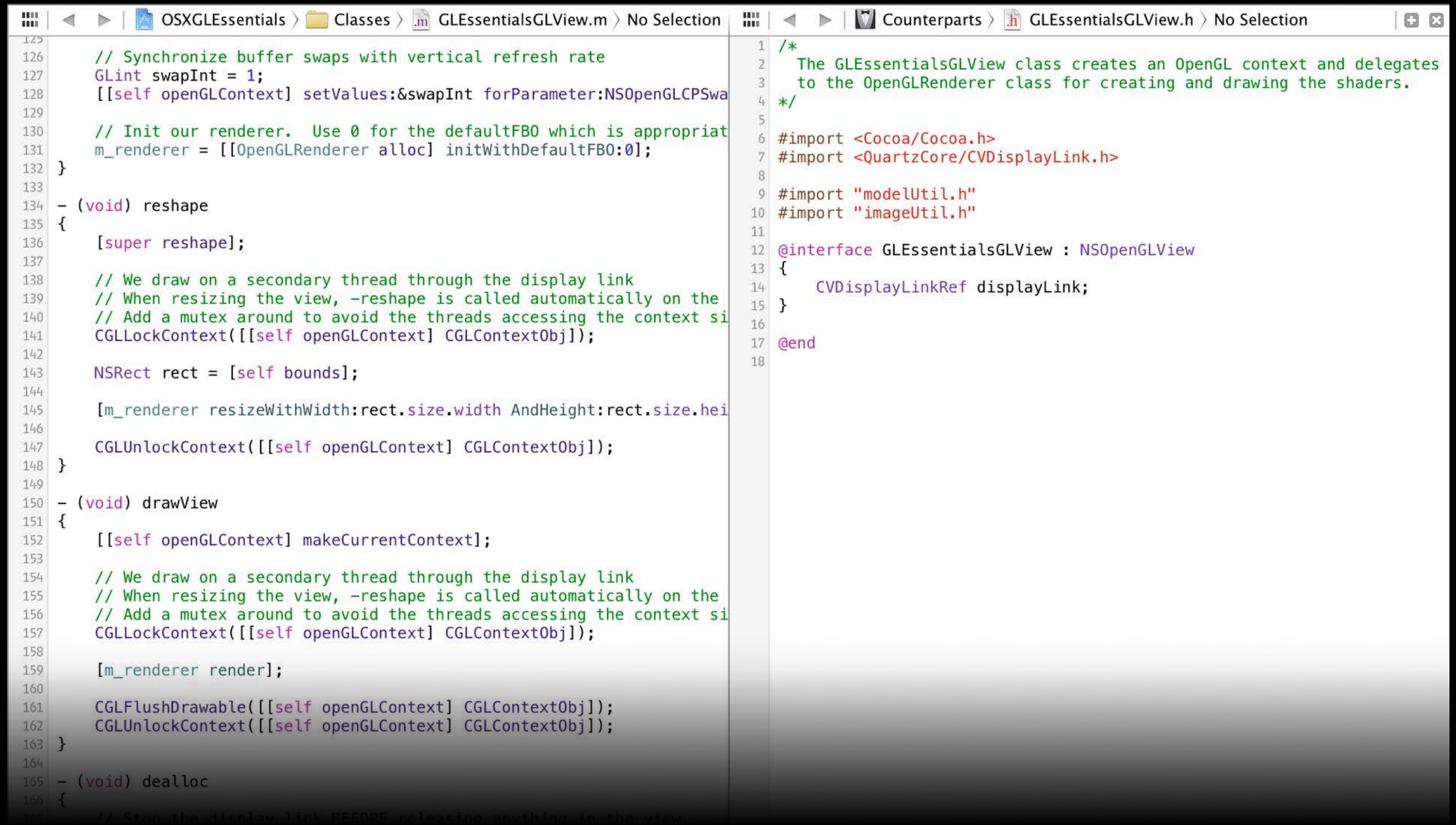
Assistant Editor



Assistant Editor



Assistant Editor



Assistant Editor

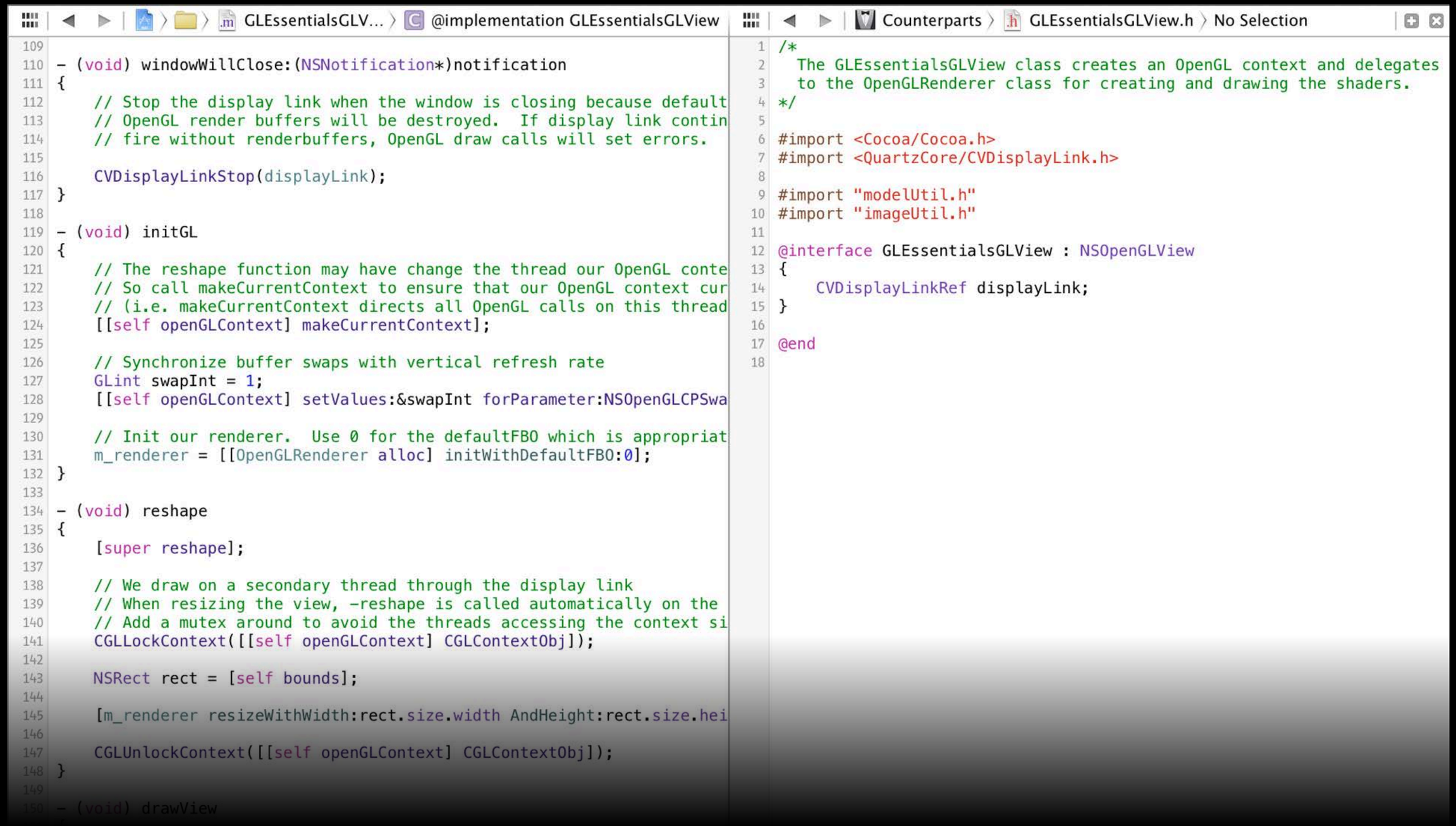
iOSGLEssentials 〉 Shaders 〉 reflect.vsh 〉 No Selection

```
/*
  The vertex shader for reflection rendering.
*/

#ifdef GL_ES
precision highp float;
#endif

uniform mat4 modelViewMatrix;
uniform mat4 modelViewProjectionMatrix;
uniform mat3 normalMatrix;

// Declare inputs and outputs
// inPosition : Position attribute from the VAO/VBOs
// inNormal : Normal attribute from the VAO/VBOs
// varNormal : Normalized normal value passed to the rasterizer and use
//             to compute the reflection texture coordinates the fragme
// verEyeDir : Direction of the eye is facing which we derive from the
//              matrix.  This is passed to the rasterizer and used to c
//              the reflection texture coordinate in the fragment shade
// gl_Position : Implicitly declared in all vertex shaders.  The clip s
//               position passed to rasterizer used to build the triang

#if __VERSION__ >= 140
in vec3  inNormal;
in vec4  inPosition;
out vec3 varNormal;
out vec3 varEyeDir;
#else
attribute vec3 inNormal;
attribute vec4 inPosition;
varying vec3  varNormal;
varying vec3  varEyeDir;
#endif

void main (void)
{
    gl_Position = modelViewProjectionMatrix * inPosition;
    vec4 eyePos = modelViewMatrix * inPosition;

    varNormal = normalize(normalMatrix * inNormal);
    varEyeDir = eyePos.xyz;
```

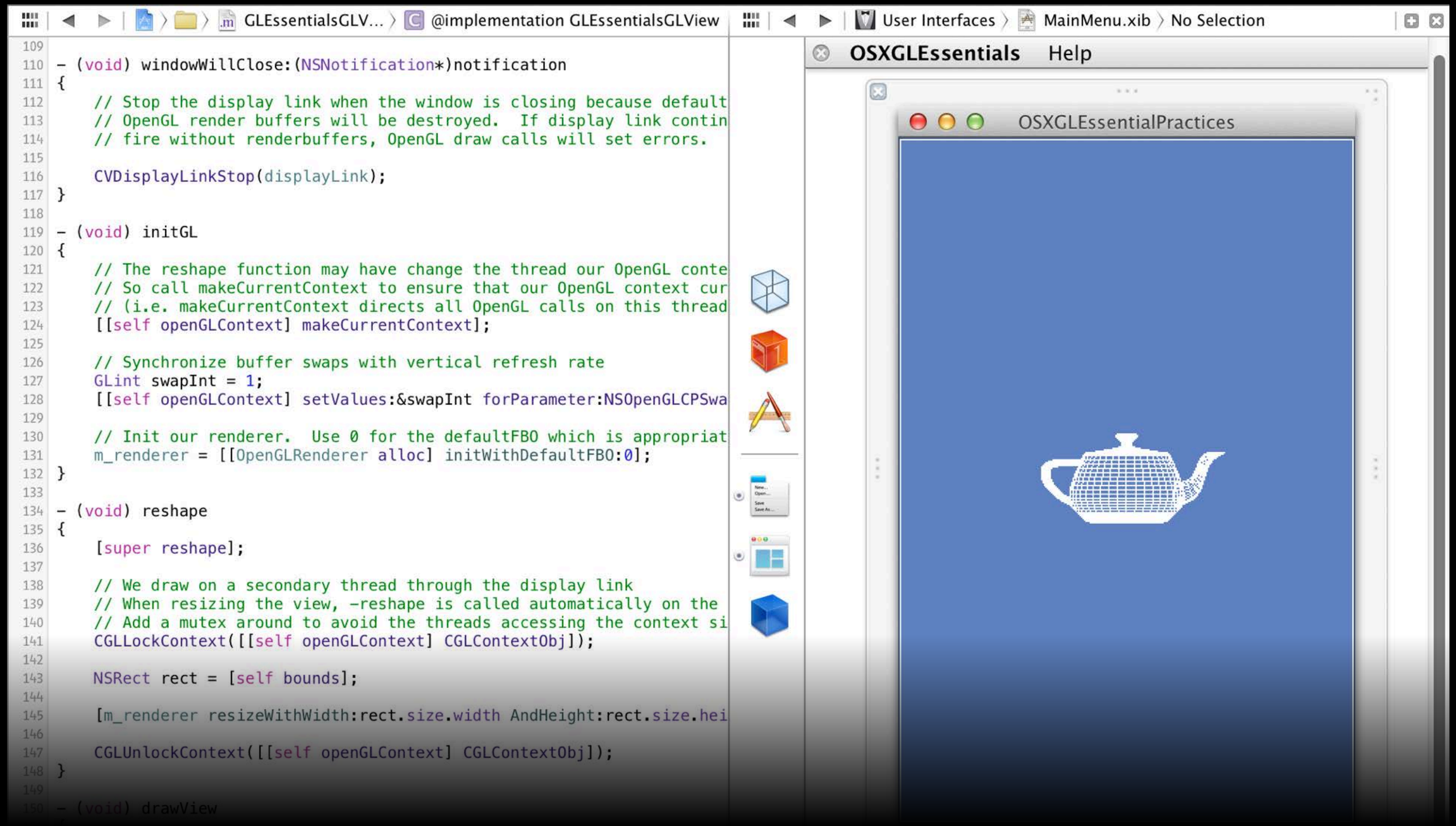
Counterparts 〉 reflect.fsh 〉 No Selection

```
/*
  The fragment shader for reflection rendering.
*/

#ifdef GL_ES
precision highp float;
#endif


const vec3 Xunitvec = vec3(1.0, 0.0, 0.0);
const vec3 Yunitvec = vec3(0.0, 1.0, 0.0);

// Color of tint to apply (blue)
const vec4 tintColor = vec4(0.0, 0.0, 1.0, 1.0);

// Amount of tint to apply
const float tintFactor = 0.2;

// Declare inputs and outputs
// varNormal : Normal for the fragment computed by the rasterizer based
//             varNormal value output in the vertex shader
// varEyeDir : EyeDir for the fragment computed by the rasterizer based
//             varEyeDir value output in the vertex shader
// gl_FragColor : Implicitly declare in fragments shaders less than 1.4
//                The output color of our fragment.
// fragColor : Output color of our fragment.  Basically the same as gl_
//             but we must explicitly declared this in shaders version
//             above.

#if __VERSION__ >= 140
in vec3       varNormal;
in vec3       varEyeDir;
out vec4      fragColor;
#else
varying vec3  varNormal;
varying vec3  varEyeDir;
#endif

uniform sampler2D diffuseTexture;

void main (void)
{
```

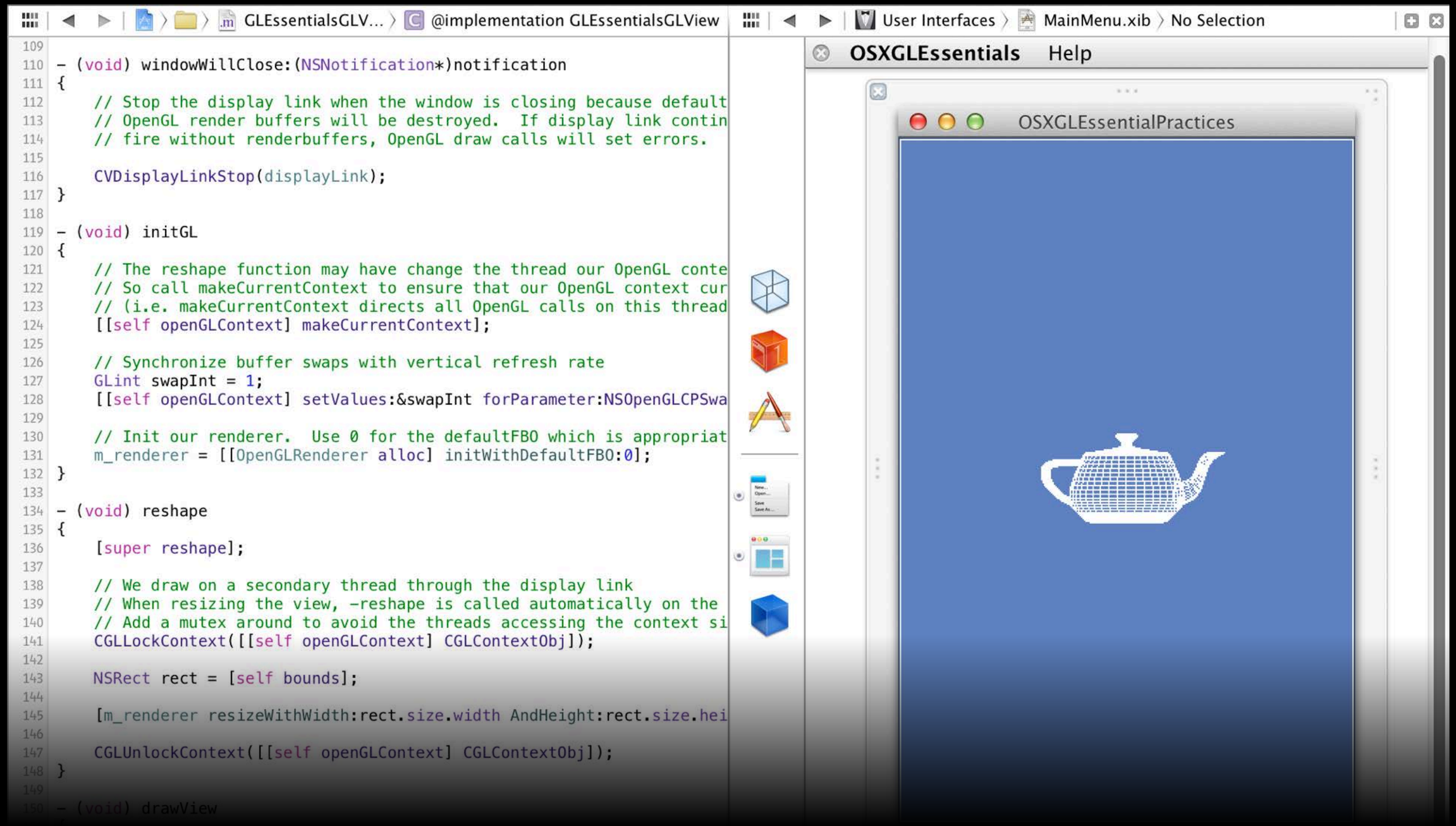
Assistant Editor



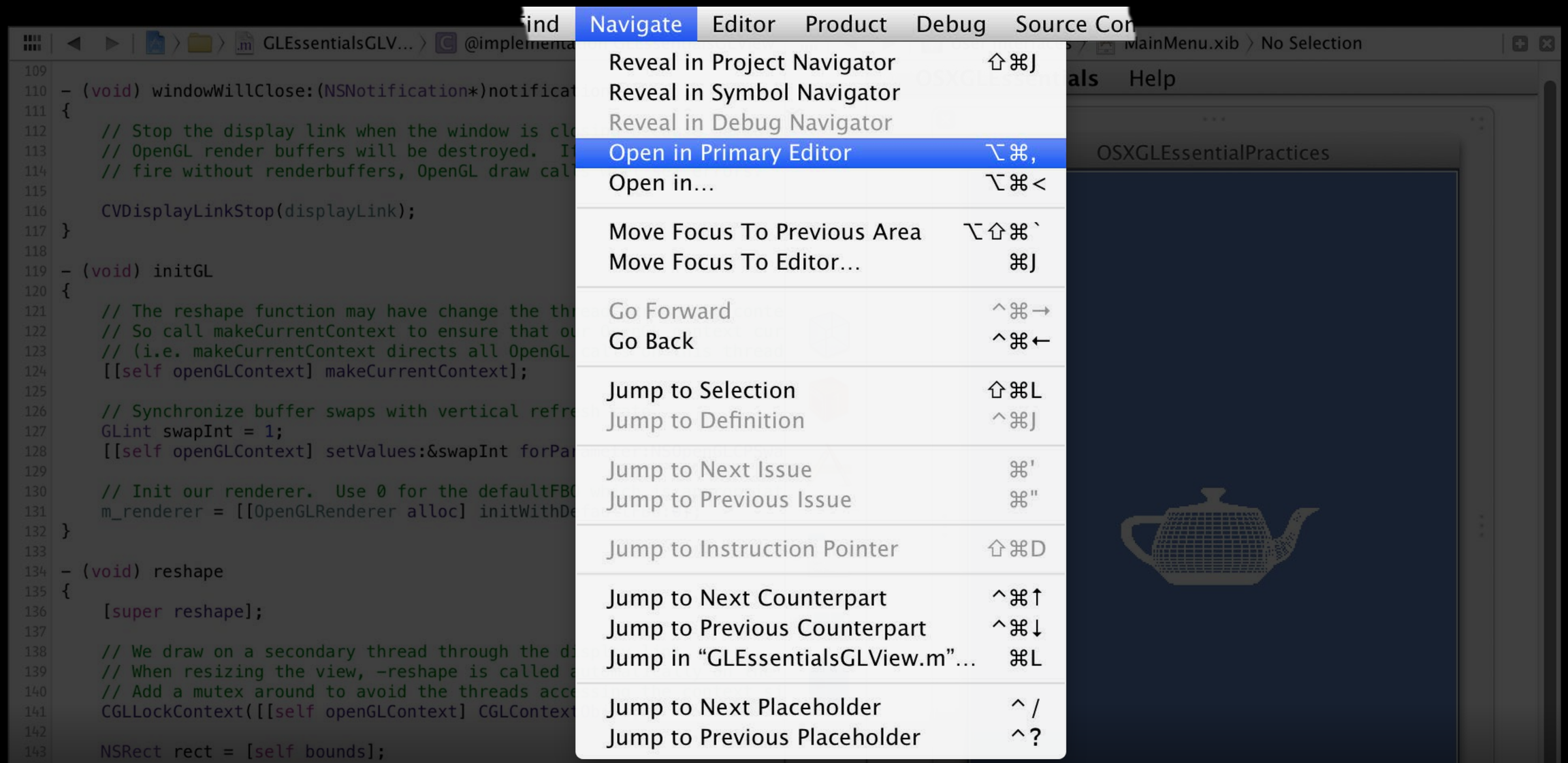
Assistant Editor



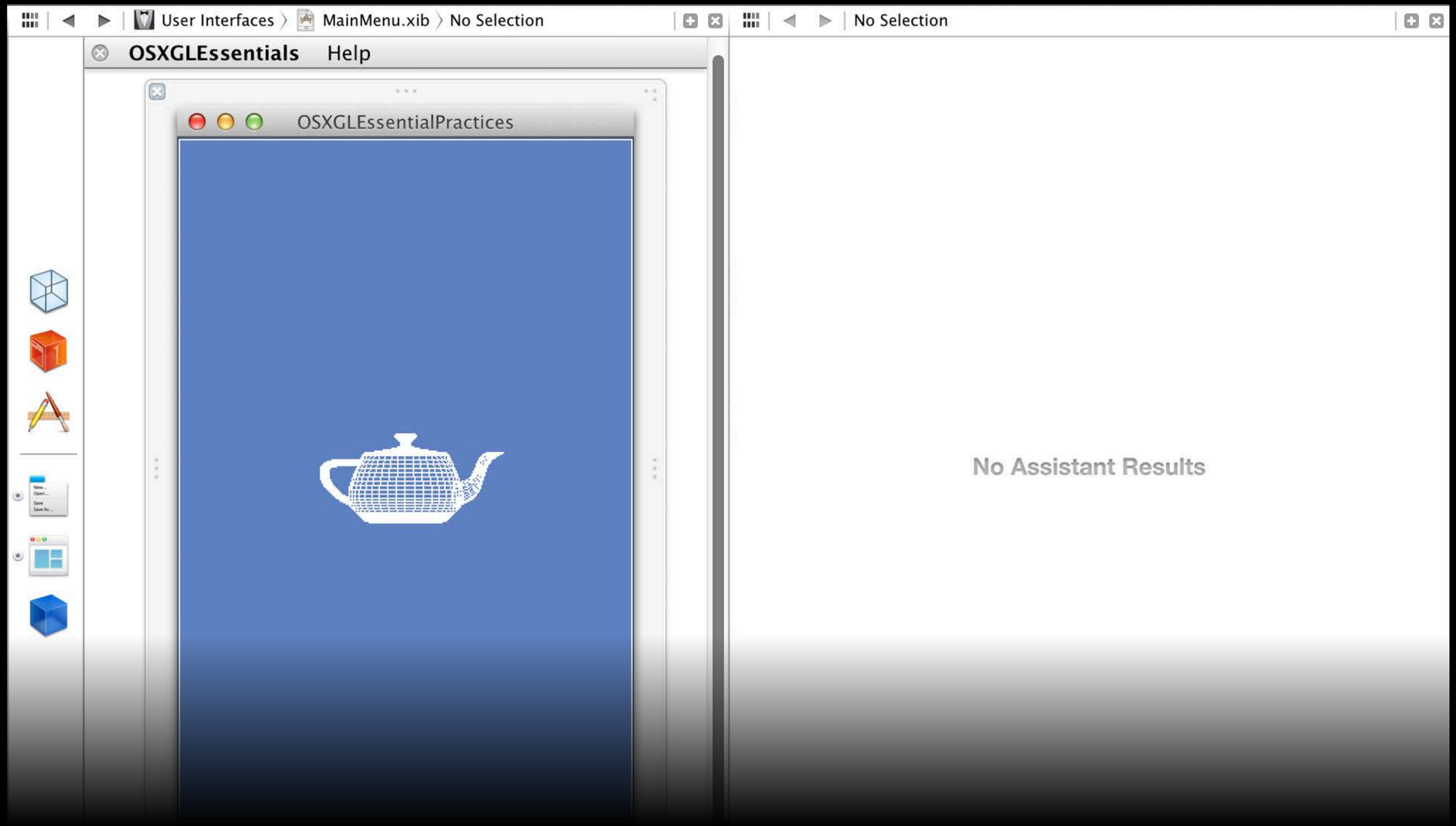
Assistant Editor



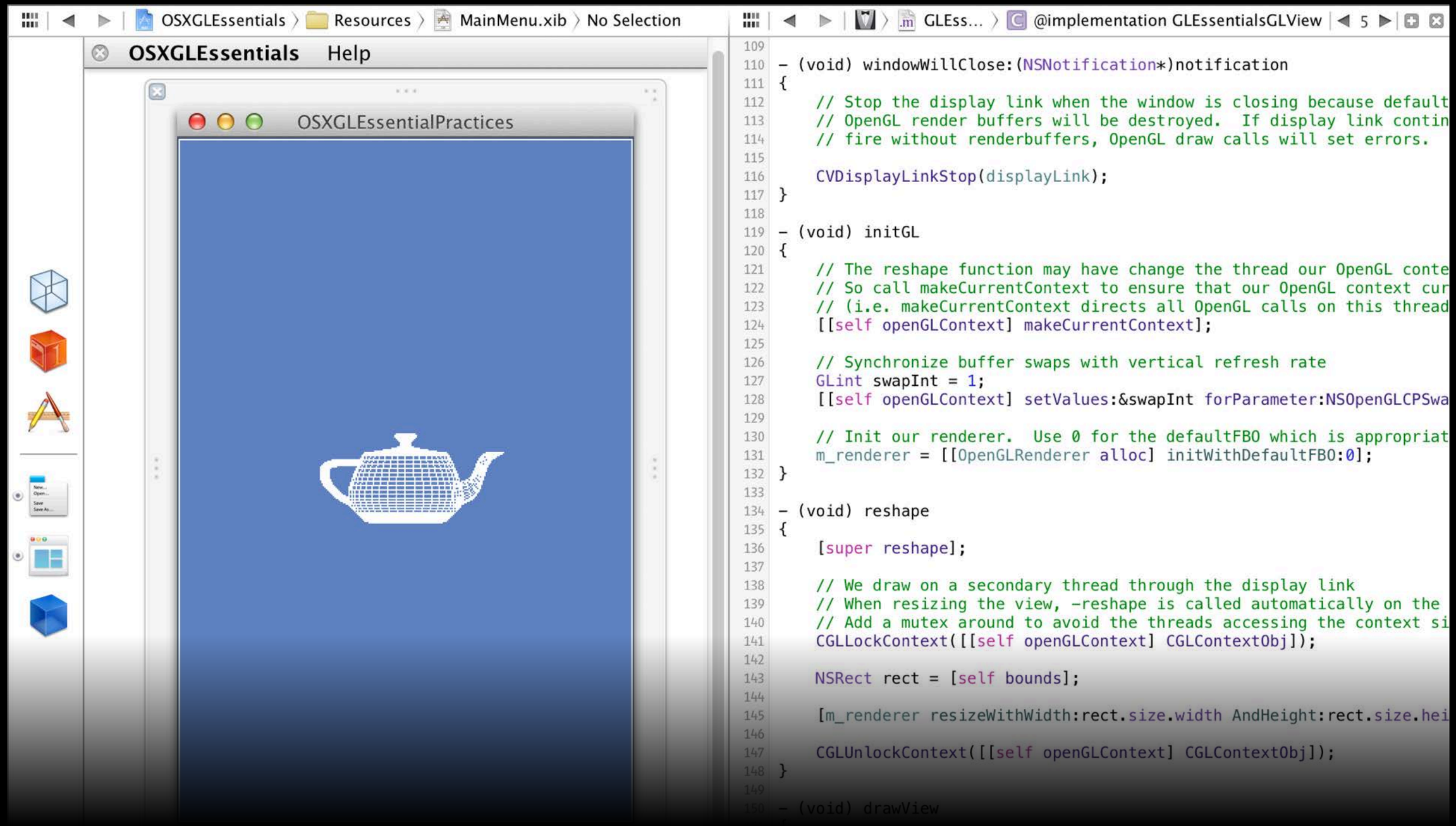
Assistant Editor



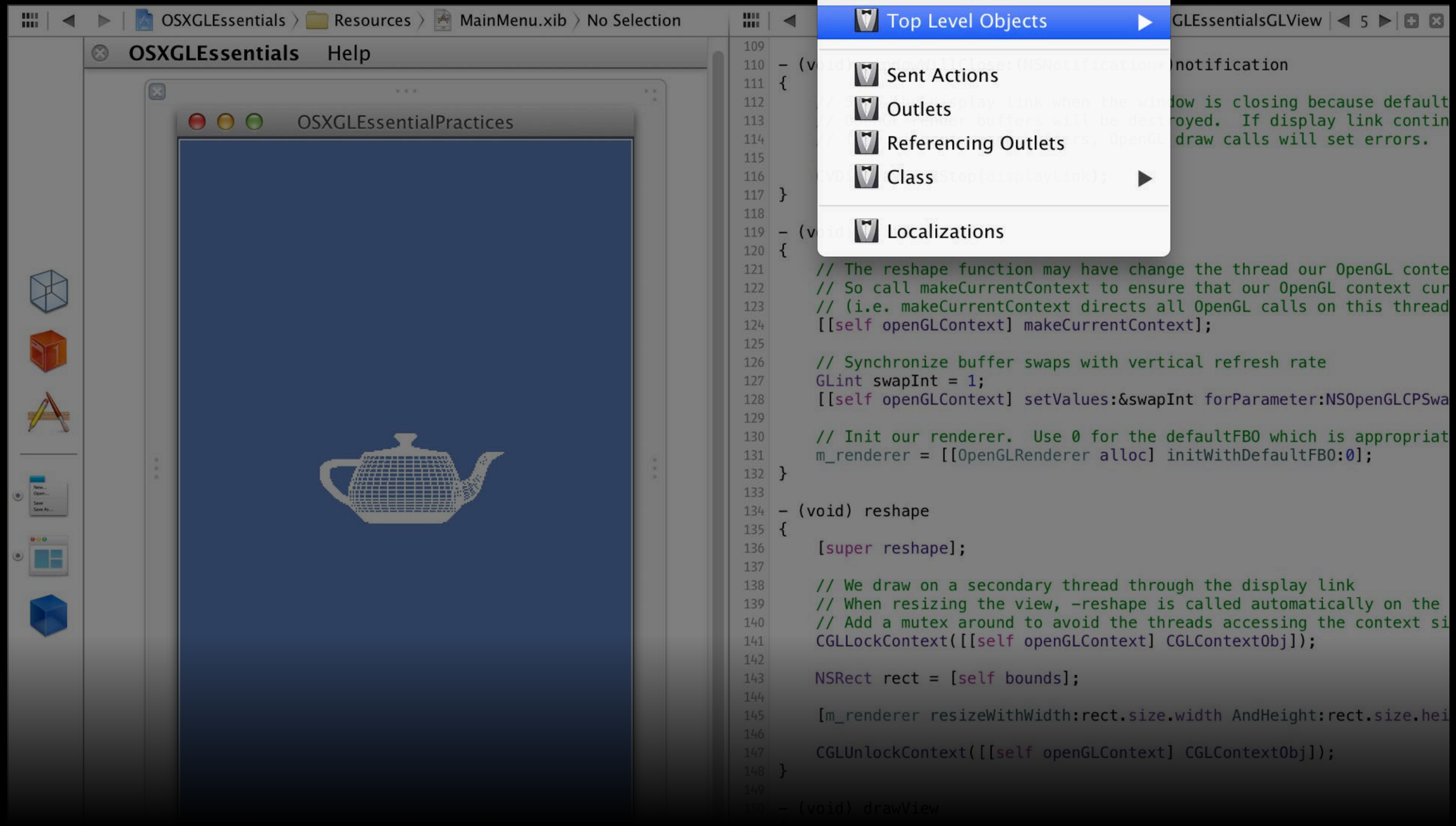
Assistant Editor



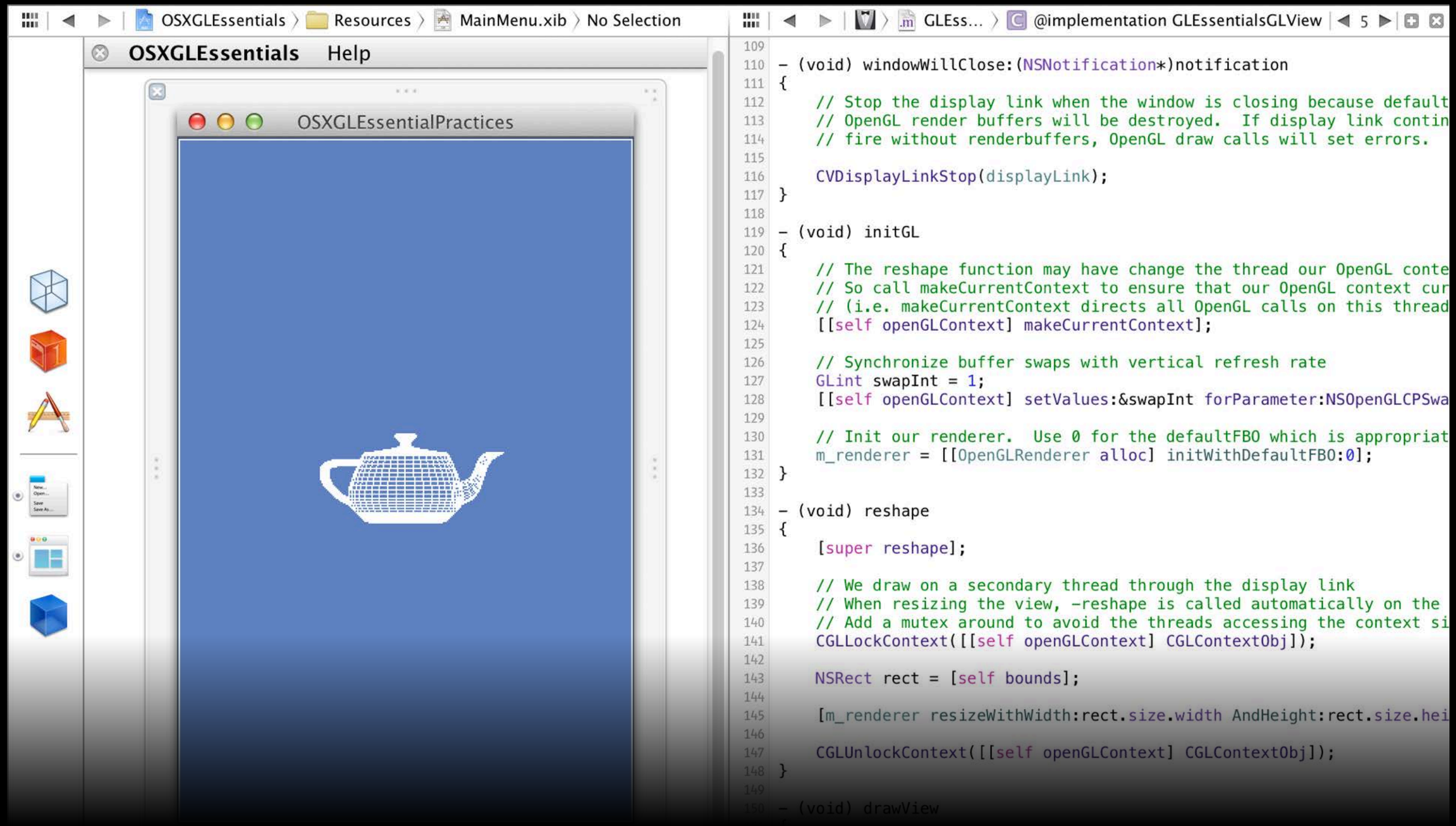
Assistant Editor



Assistant Editor



Assistant Editor



Activating the Assistant Editor

Activating the Assistant Editor

Toolbar button

Activating the Assistant Editor

Toolbar button

Option key

The Option Key

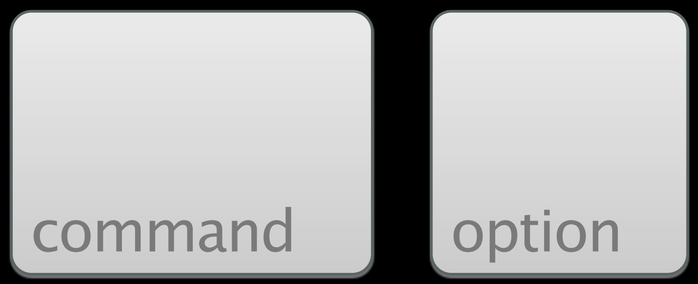
The Option Key

The Option Key

Jump to definition
in focused editor

The Option Key

Jump to definition
in focused editor

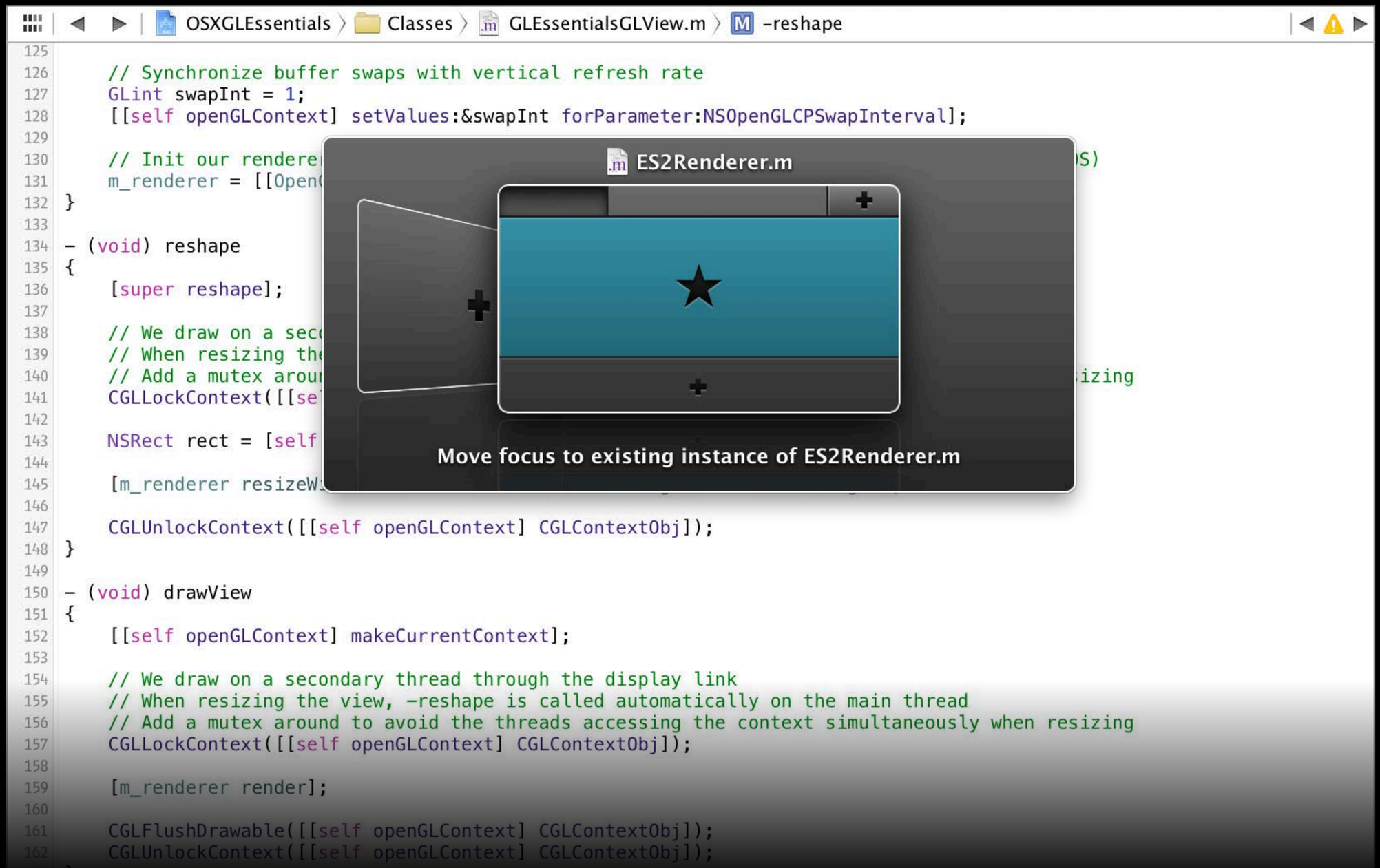


The Option Key

Jump to definition
in focused editor

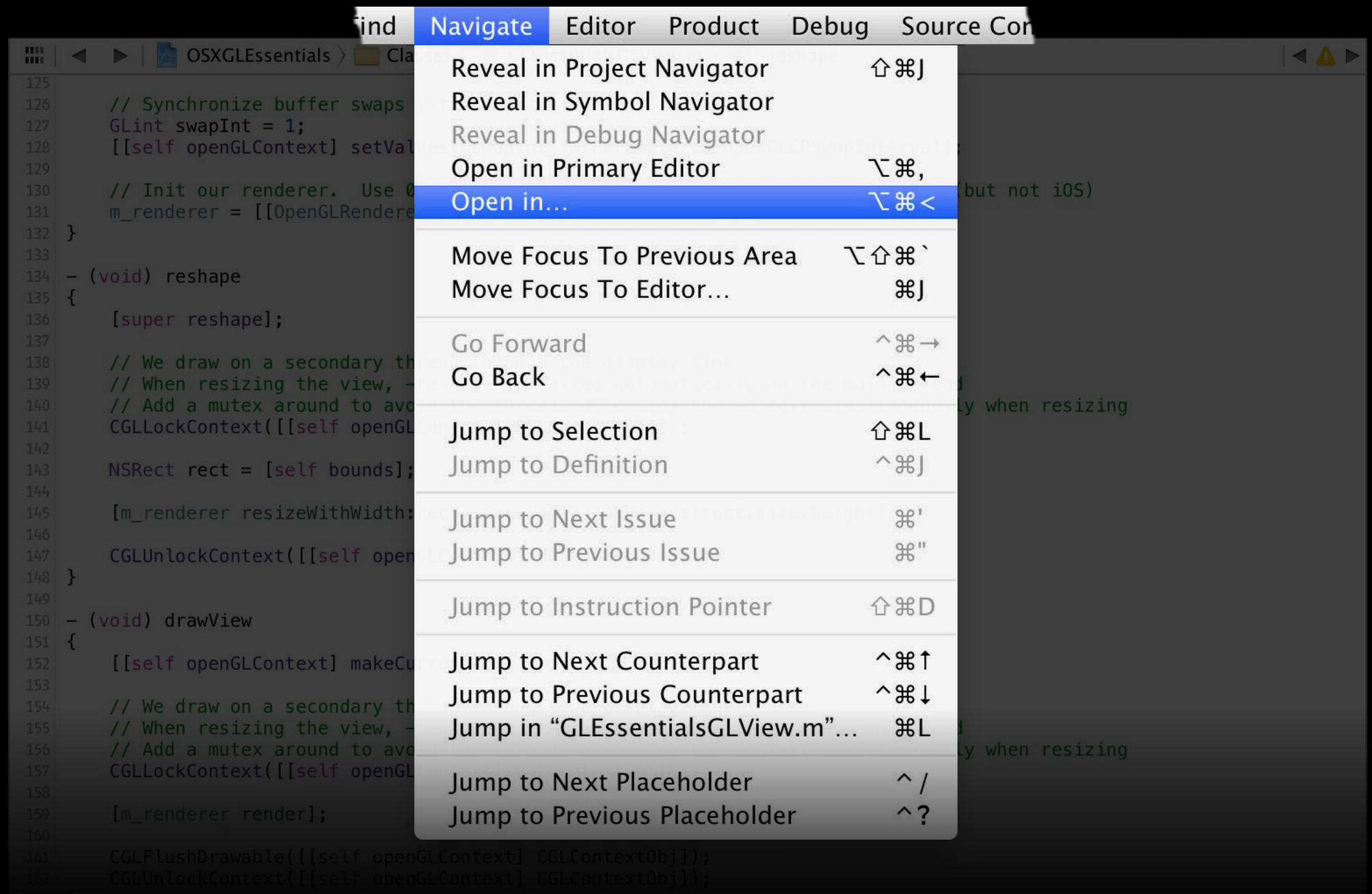
Jump to definition
in assistant editor

Navigation Chooser

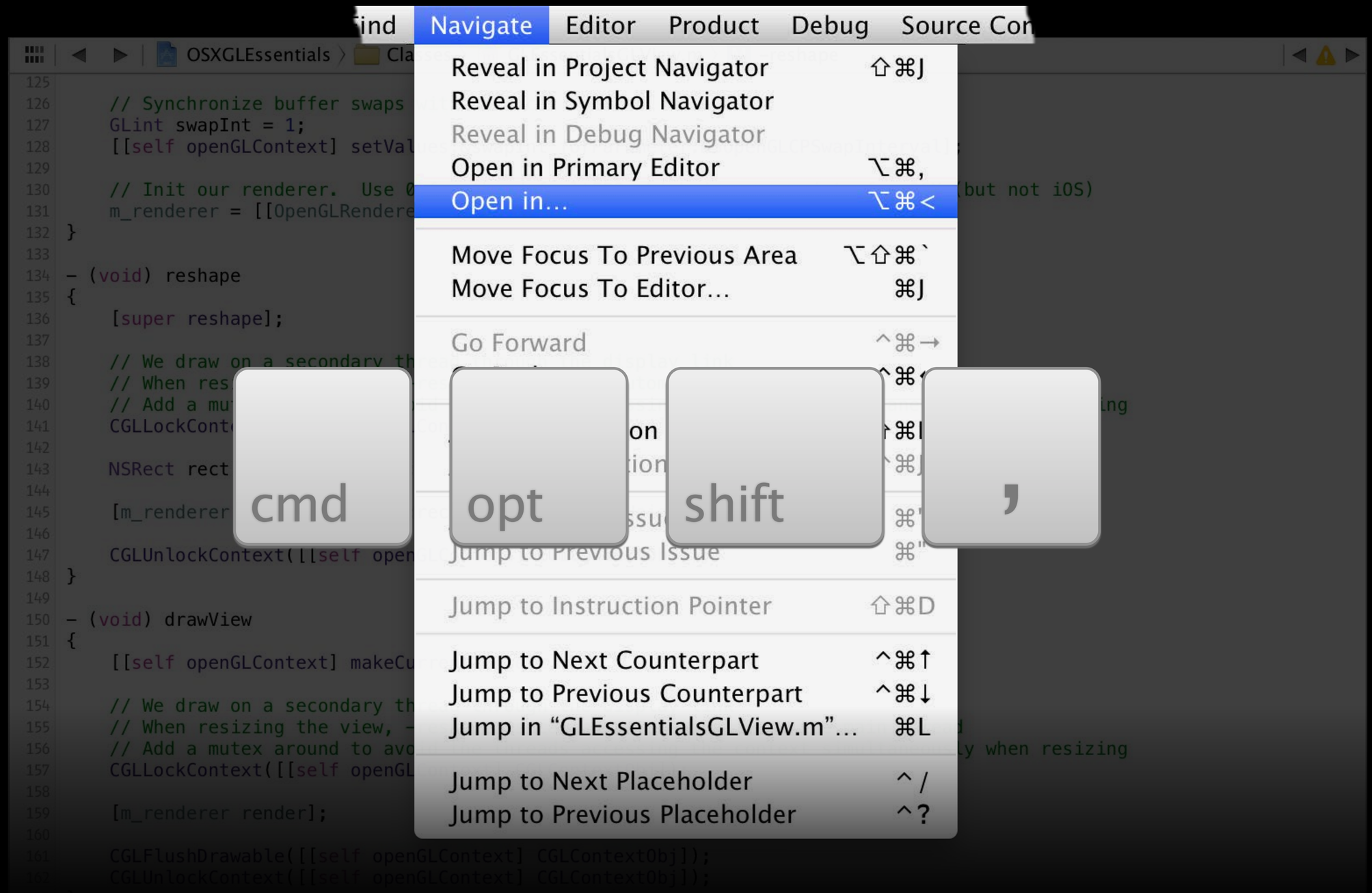



```
125
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129
130     // Init our rendere                                                    S)
131     m_renderer = [[Open(
132 }
133
134 - (void) reshape
135 {
136     [super reshape];
137
138     // We draw on a sec(
139     // When resizing the
140     // Add a mutex arou                                                   izing
141     CGLLockContext([[se
142
143     NSRect rect = [self
144
145     [m_renderer resizeW
146
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
149
150 - (void) drawView
151 {
152     [[self openGLContext] makeCurrentContext];
153
154     // We draw on a secondary thread through the display link
155     // When resizing the view, -reshape is called automatically on the main thread
156     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
157     CGLLockContext([[self openGLContext] CGLContextObj]);
158
159     [m_renderer render];
160
161     CGLFlushDrawable([[self openGLContext] CGLContextObj]);
162     CGLUnlockContext([[self openGLContext] CGLContextObj]);
163 }
```

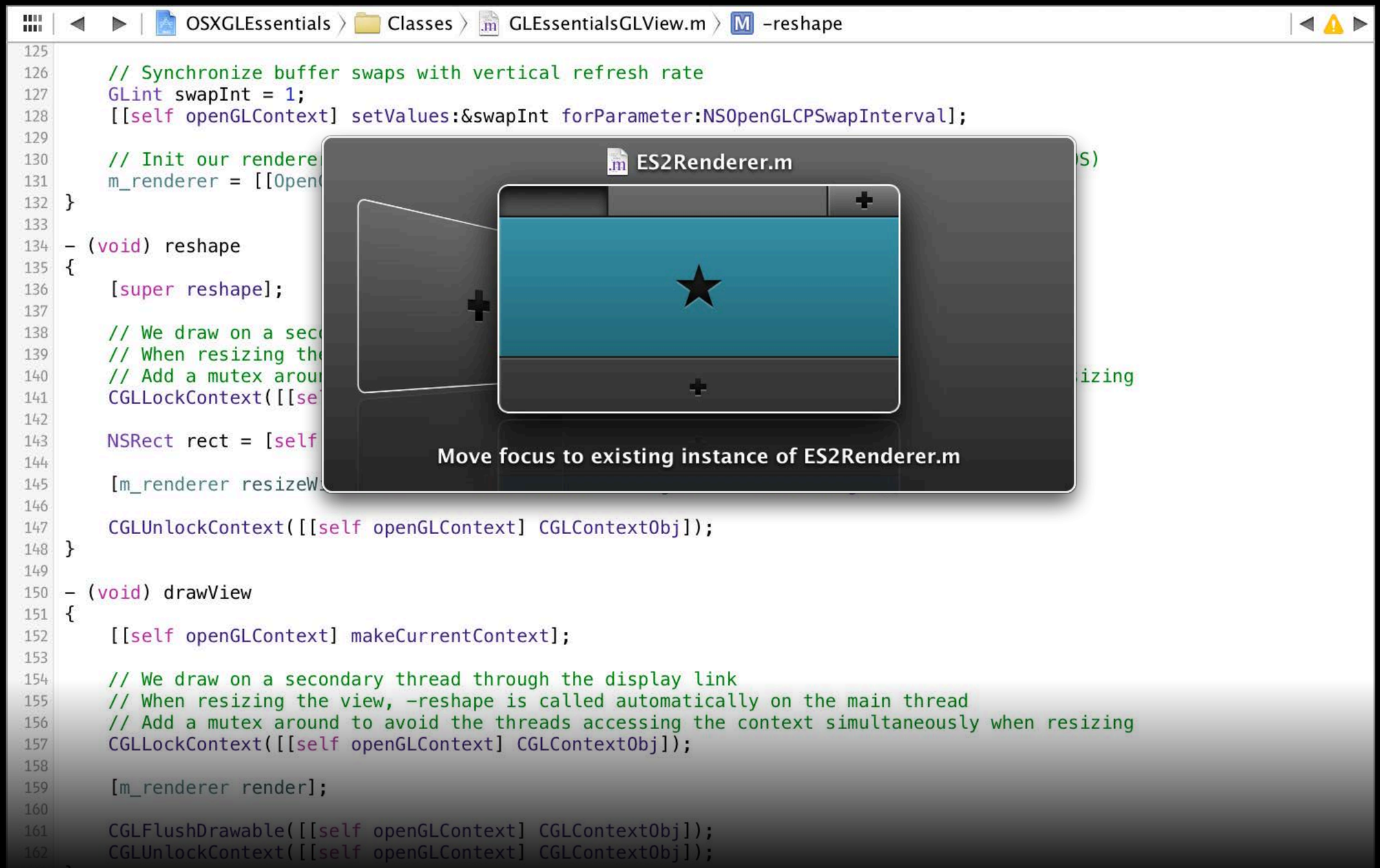
Navigation Chooser



Navigation Chooser

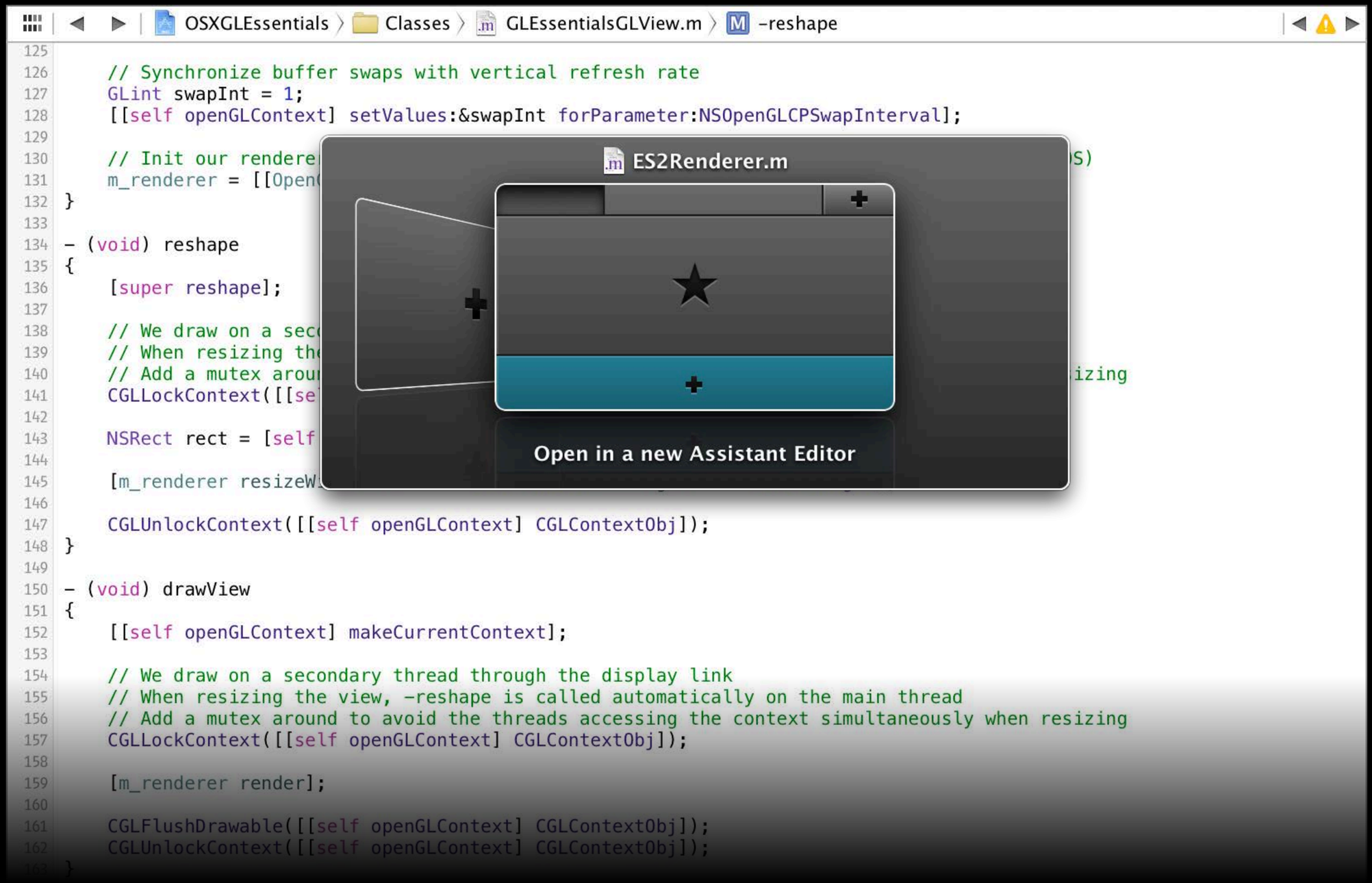


Navigation Chooser



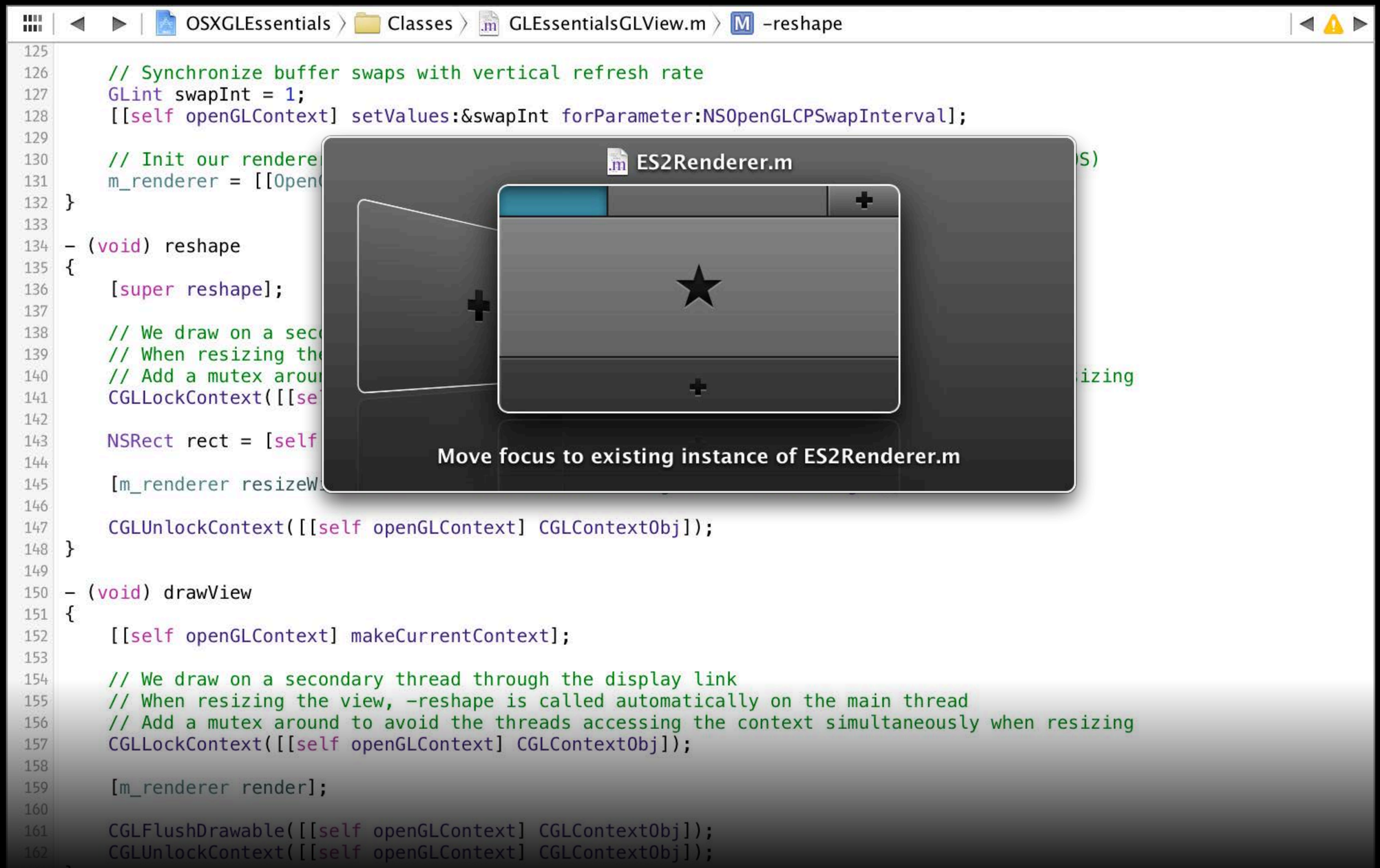

```
125
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129
130     // Init our rendere                                                    S)
131     m_renderer = [[Open(
132 }
133
134 - (void) reshape
135 {
136     [super reshape];
137
138     // We draw on a sec(
139     // When resizing th(
140     // Add a mutex arou                                                   izing
141     CGLLockContext([[se
142
143     NSRect rect = [self
144
145     [m_renderer resizeW
146
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
149
150 - (void) drawView
151 {
152     [[self openGLContext] makeCurrentContext];
153
154     // We draw on a secondary thread through the display link
155     // When resizing the view, -reshape is called automatically on the main thread
156     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
157     CGLLockContext([[self openGLContext] CGLContextObj]);
158
159     [m_renderer render];
160
161     CGLFlushDrawable([[self openGLContext] CGLContextObj]);
162     CGLUnlockContext([[self openGLContext] CGLContextObj]);
163 }
```

Navigation Chooser



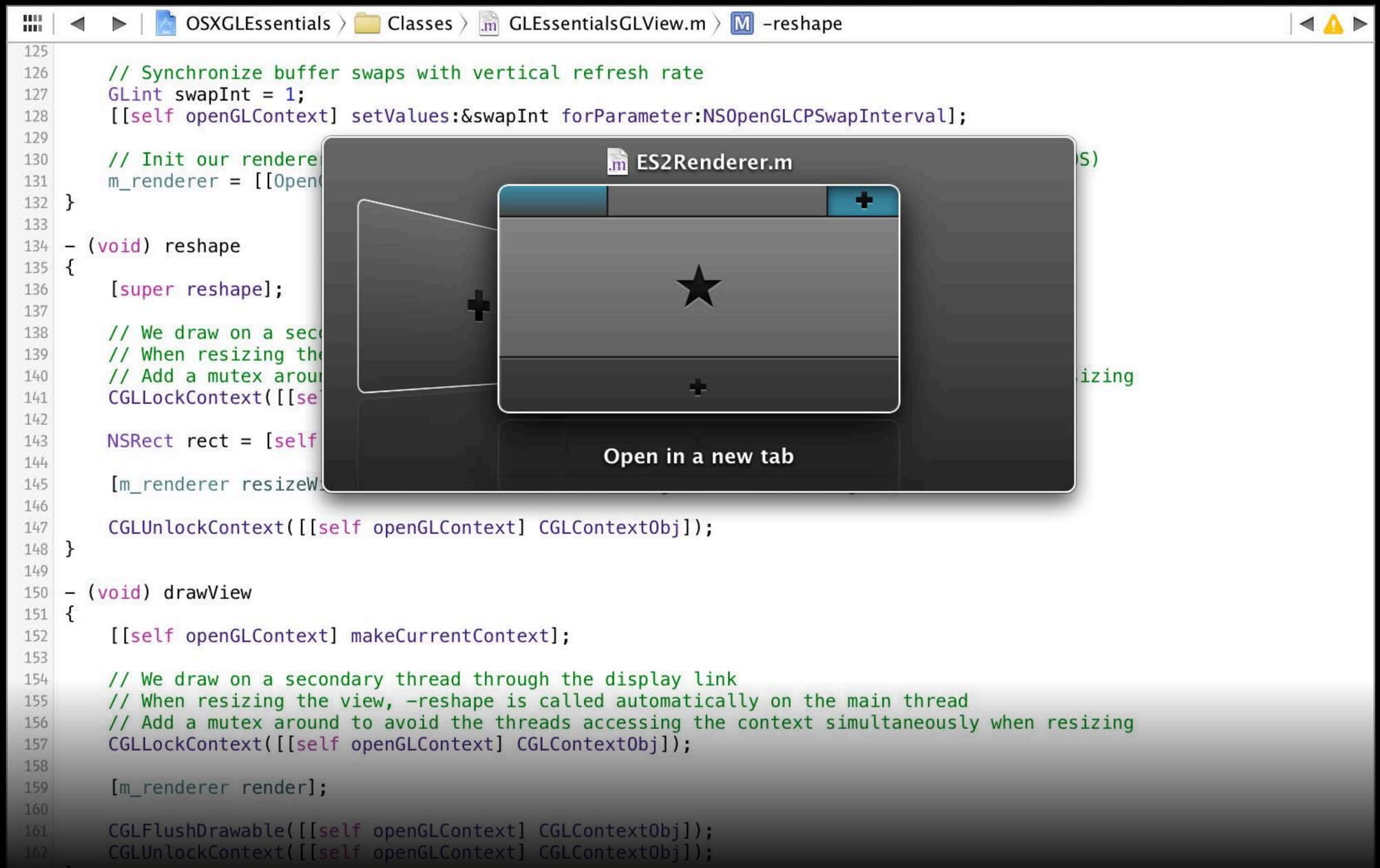

```
125
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129
130     // Init our rendere                                                    S)
131     m_renderer = [[Open(
132 }
133
134 - (void) reshape
135 {
136     [super reshape];
137
138     // We draw on a sec(
139     // When resizing th(
140     // Add a mutex arou                                                   izing
141     CGLLockContext([[se
142
143     NSRect rect = [self
144
145     [m_renderer resizeW
146
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
149
150 - (void) drawView
151 {
152     [[self openGLContext] makeCurrentContext];
153
154     // We draw on a secondary thread through the display link
155     // When resizing the view, -reshape is called automatically on the main thread
156     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
157     CGLLockContext([[self openGLContext] CGLContextObj]);
158
159     [m_renderer render];
160
161     CGLFlushDrawable([[self openGLContext] CGLContextObj]);
162     CGLUnlockContext([[self openGLContext] CGLContextObj]);
163 }
```

Navigation Chooser



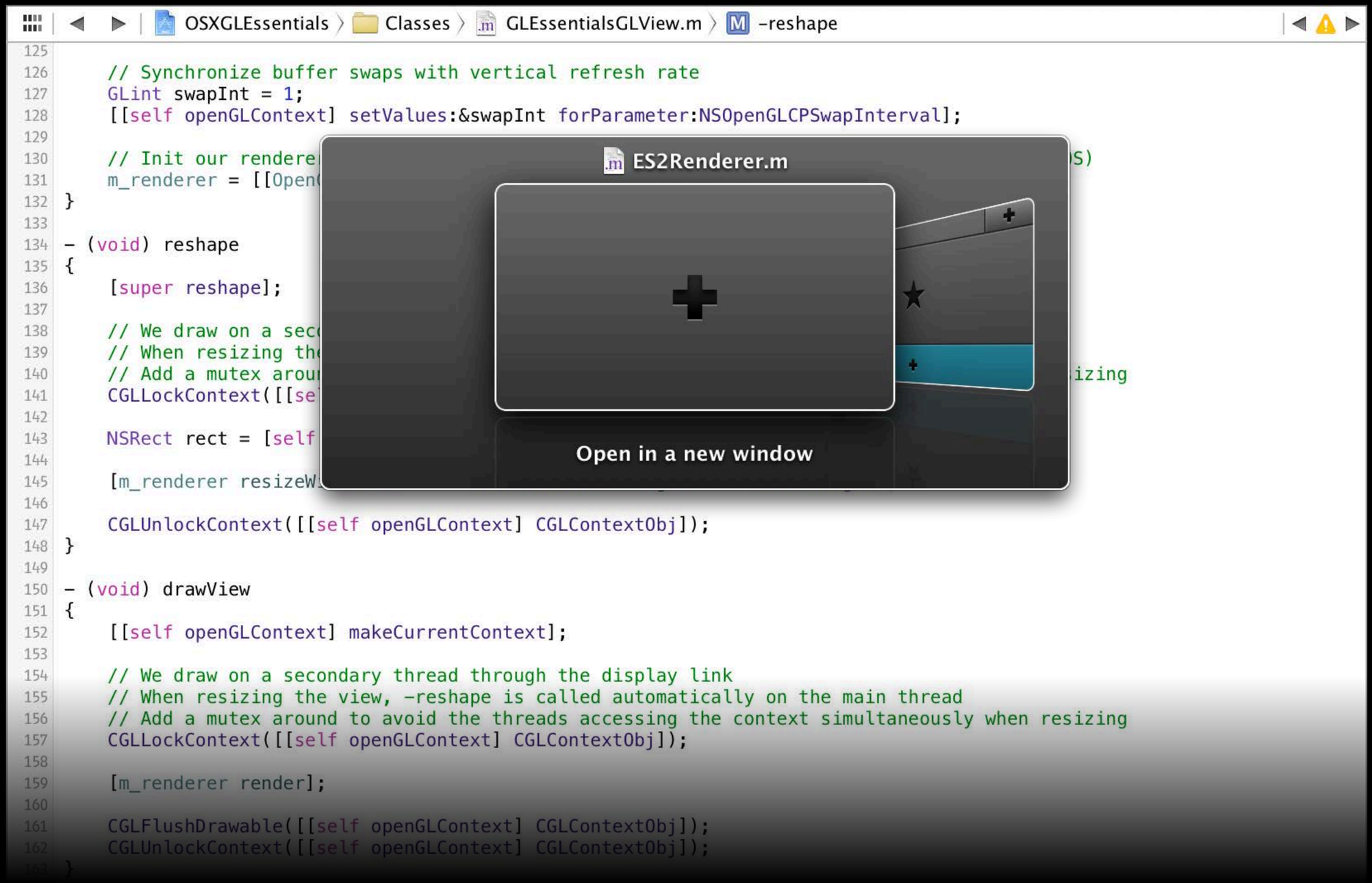

```
125
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129
130     // Init our rendere                                                    S)
131     m_renderer = [[Open(
132 }
133
134 - (void) reshape
135 {
136     [super reshape];
137
138     // We draw on a sec(
139     // When resizing th(
140     // Add a mutex arou                                                   izing
141     CGLLockContext([[se
142
143     NSRect rect = [self
144
145     [m_renderer resizeW
146
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
149
150 - (void) drawView
151 {
152     [[self openGLContext] makeCurrentContext];
153
154     // We draw on a secondary thread through the display link
155     // When resizing the view, -reshape is called automatically on the main thread
156     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
157     CGLLockContext([[self openGLContext] CGLContextObj]);
158
159     [m_renderer render];
160
161     CGLFlushDrawable([[self openGLContext] CGLContextObj]);
162     CGLUnlockContext([[self openGLContext] CGLContextObj]);
163 }
```

Navigation Chooser




```
125
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129
130     // Init our rendere                                                    S)
131     m_renderer = [[Open(
132 }
133
134 - (void) reshape
135 {
136     [super reshape];
137
138     // We draw on a sec(
139     // When resizing th(
140     // Add a mutex arou                                                   izing
141     CGLLockContext([[se
142
143     NSRect rect = [self
144
145     [m_renderer resizeW
146
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
149
150 - (void) drawView
151 {
152     [[self openGLContext] makeCurrentContext];
153
154     // We draw on a secondary thread through the display link
155     // When resizing the view, -reshape is called automatically on the main thread
156     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
157     CGLLockContext([[self openGLContext] CGLContextObj]);
158
159     [m_renderer render];
160
161     CGLFlushDrawable([[self openGLContext] CGLContextObj]);
162     CGLUnlockContext([[self openGLContext] CGLContextObj]);
163 }
```

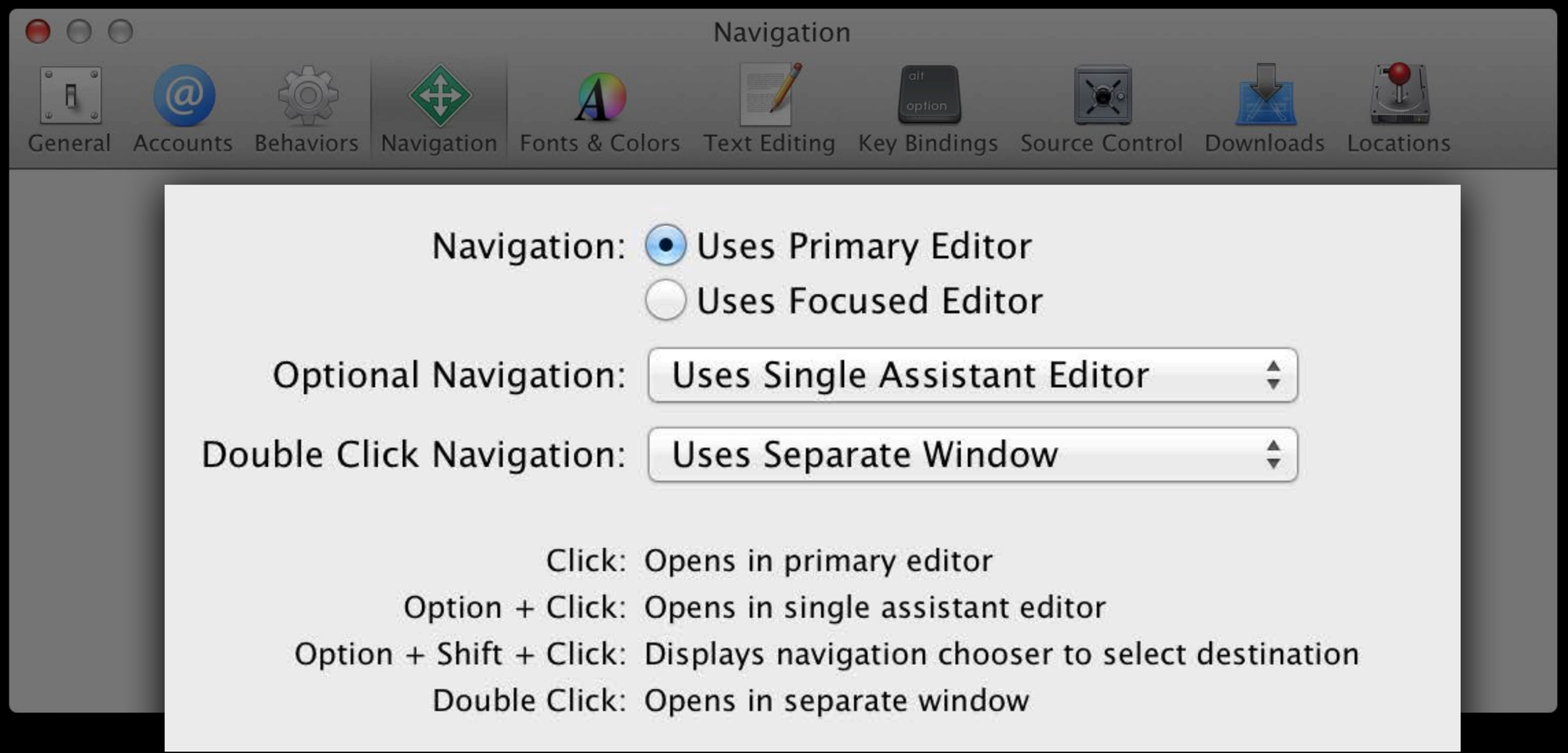
Navigation Chooser




```
125
126     // Synchronize buffer swaps with vertical refresh rate
127     GLint swapInt = 1;
128     [[self openGLContext] setValues:&swapInt forParameter:NSOpenGLCPSwapInterval];
129
130     // Init our rendere                                                    S)
131     m_renderer = [[Open(
132 }
133
134 - (void) reshape
135 {
136     [super reshape];
137
138     // We draw on a sec(
139     // When resizing th(
140     // Add a mutex arou                                                    izing
141     CGLLockContext([[se
142
143     NSRect rect = [self
144
145     [m_renderer resizeW
146
147     CGLUnlockContext([[self openGLContext] CGLContextObj]);
148 }
149
150 - (void) drawView
151 {
152     [[self openGLContext] makeCurrentContext];
153
154     // We draw on a secondary thread through the display link
155     // When resizing the view, -reshape is called automatically on the main thread
156     // Add a mutex around to avoid the threads accessing the context simultaneously when resizing
157     CGLLockContext([[self openGLContext] CGLContextObj]);
158
159     [m_renderer render];
160
161     CGLFlushDrawable([[self openGLContext] CGLContextObj]);
162     CGLUnlockContext([[self openGLContext] CGLContextObj]);
163 }
```

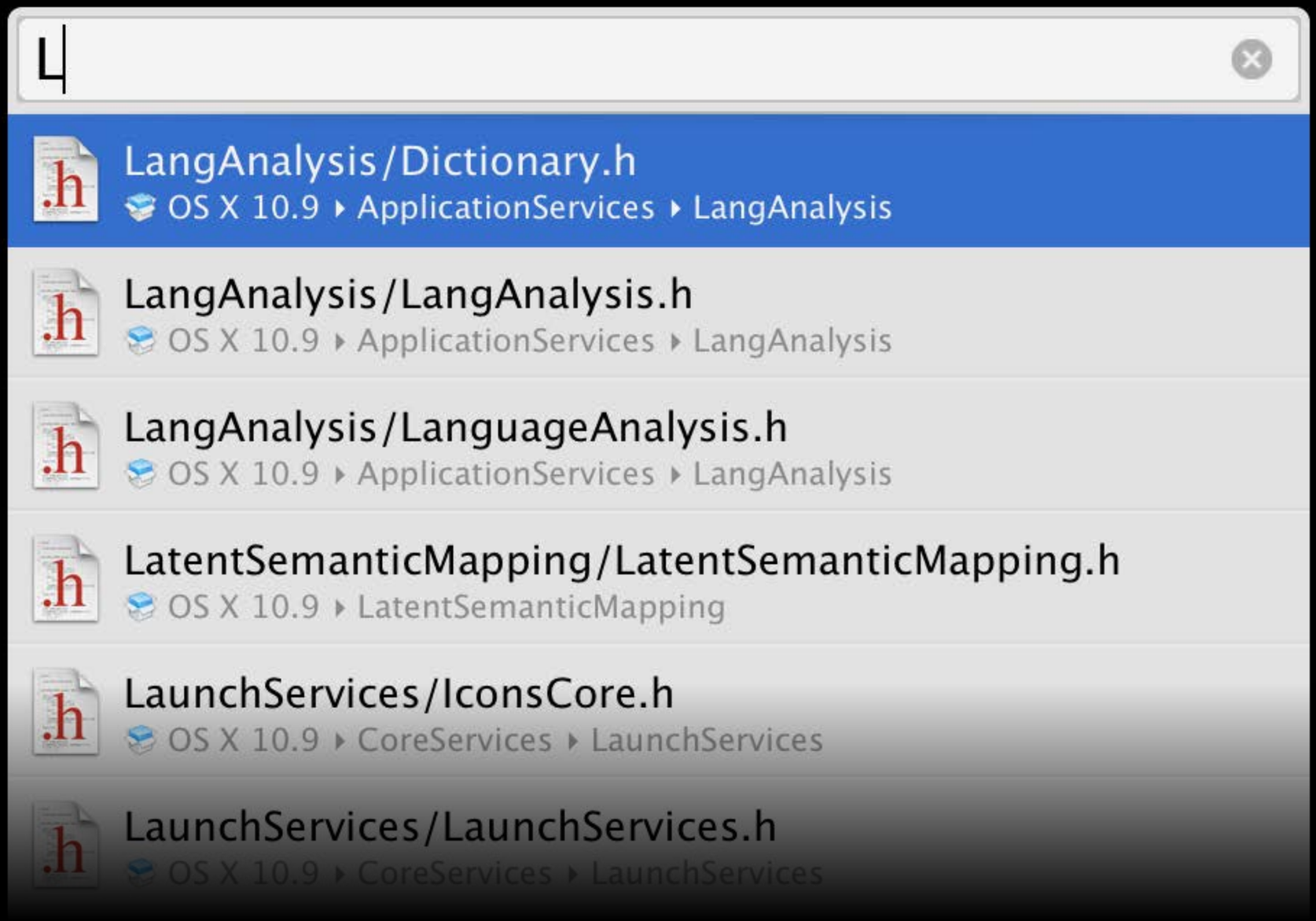
Modified Navigation

Modified Navigation



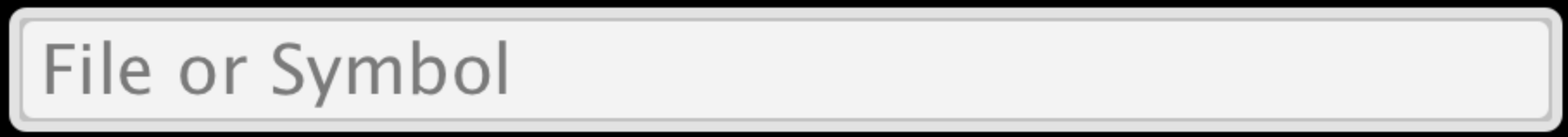
Modified Navigation

Open Quickly

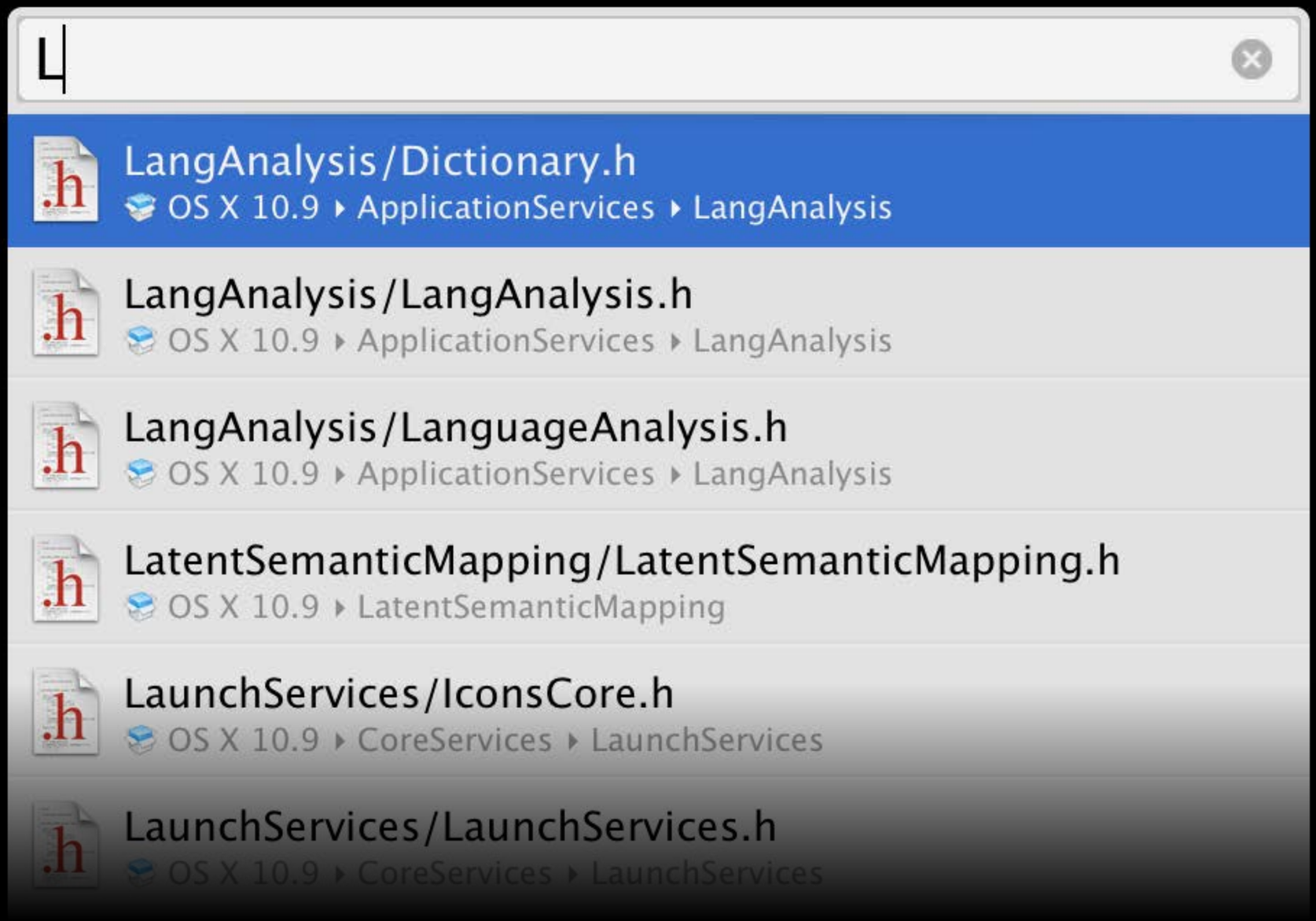


Open Quickly

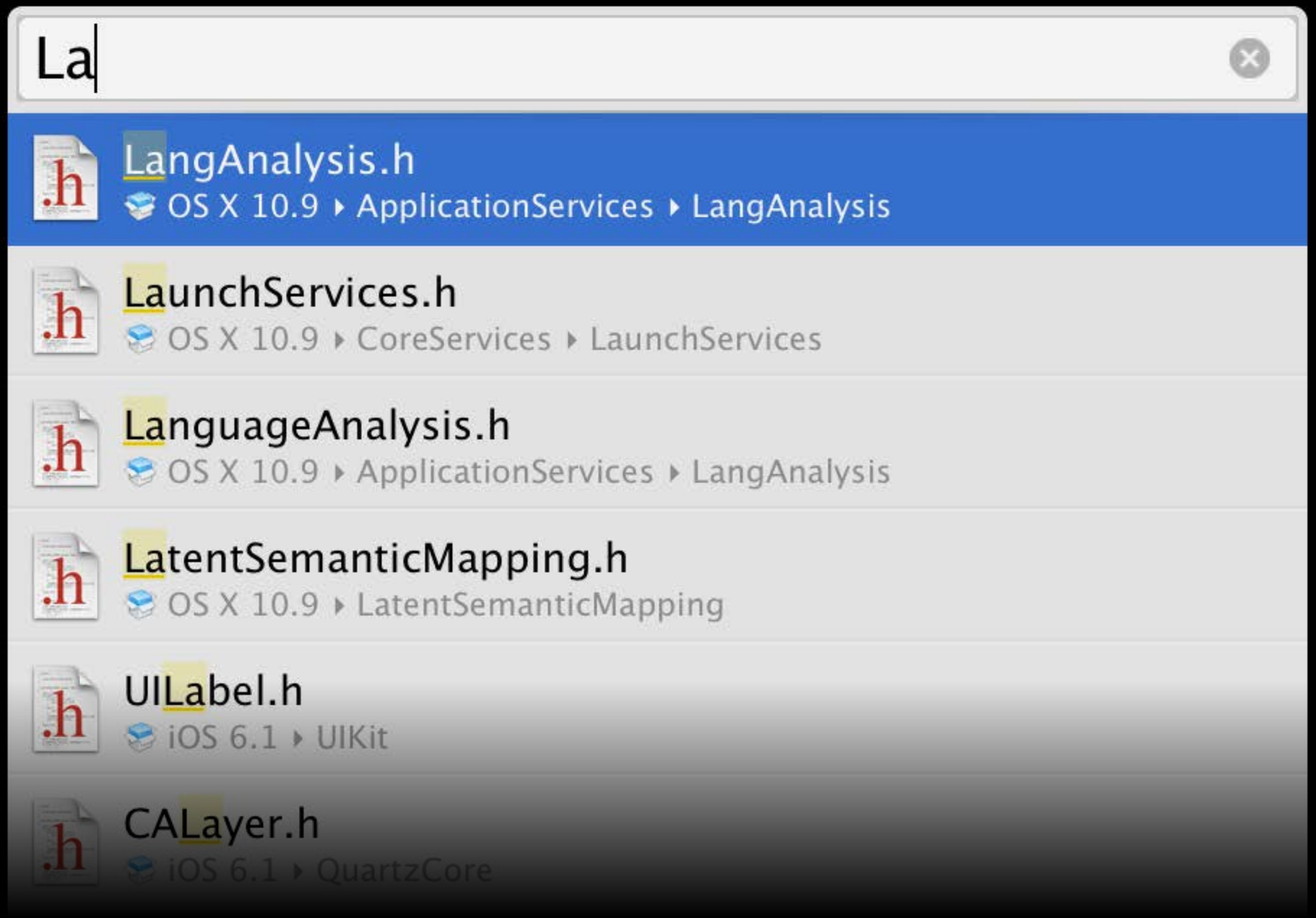
Open Quickly



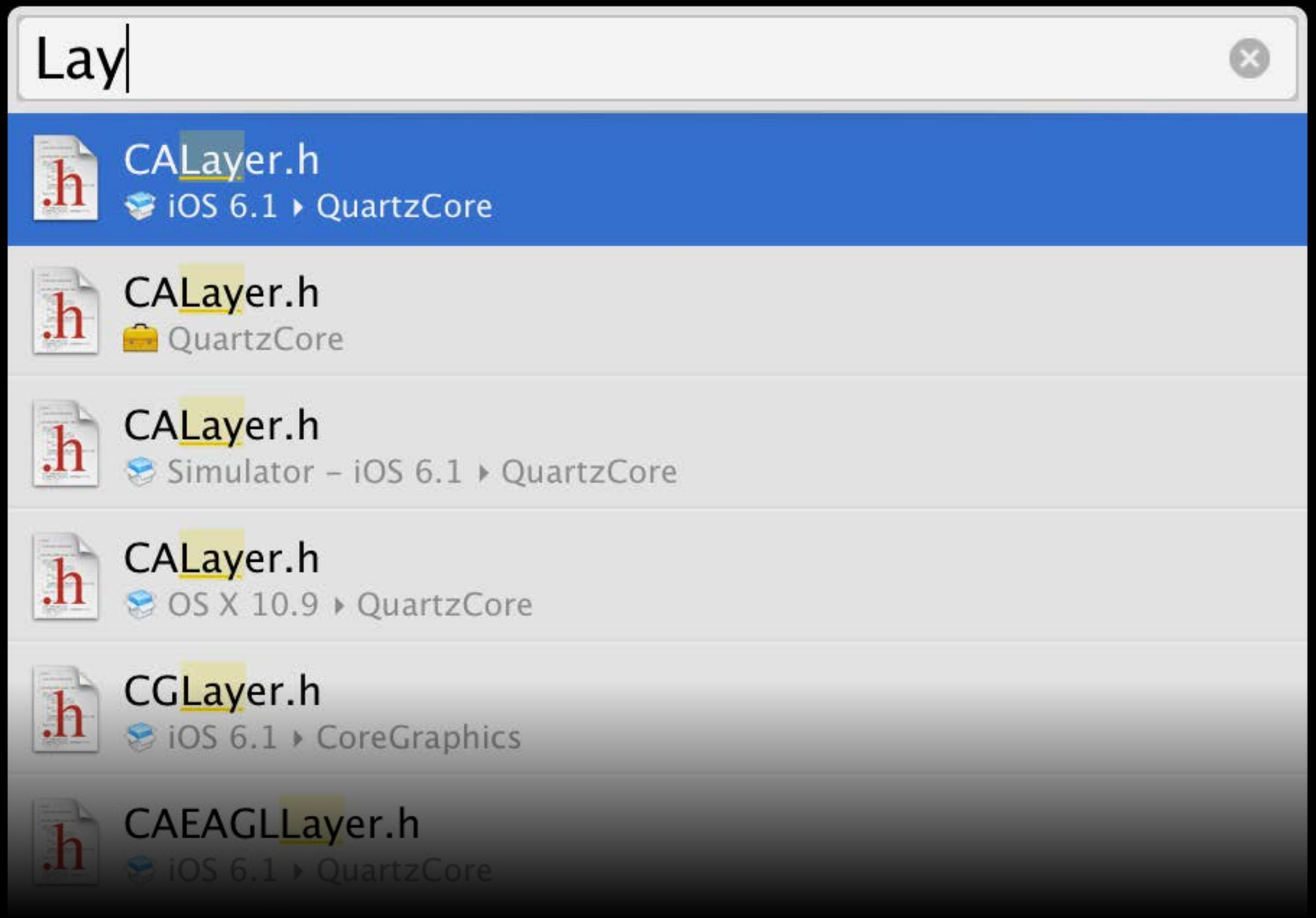
Open Quickly



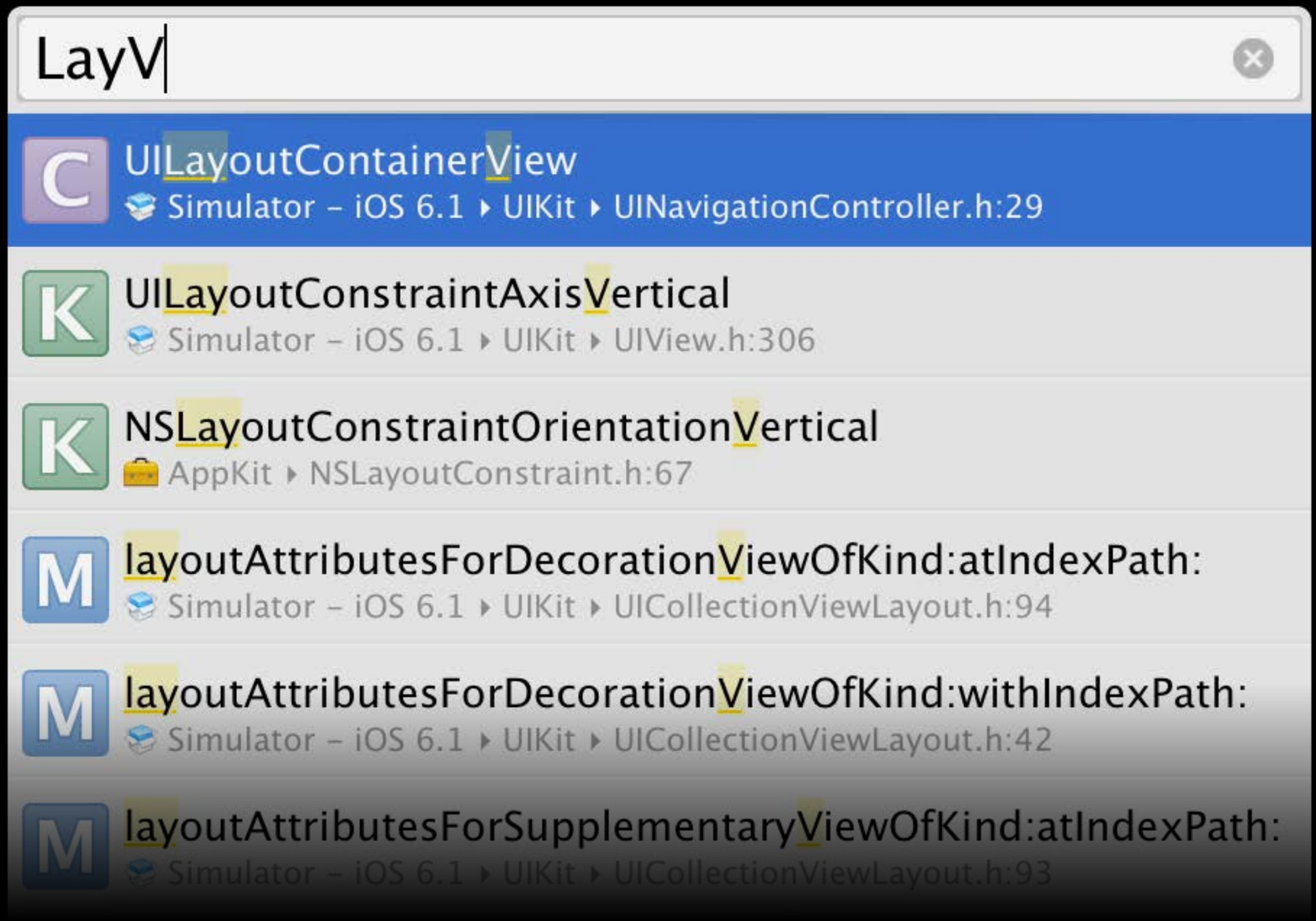
Open Quickly



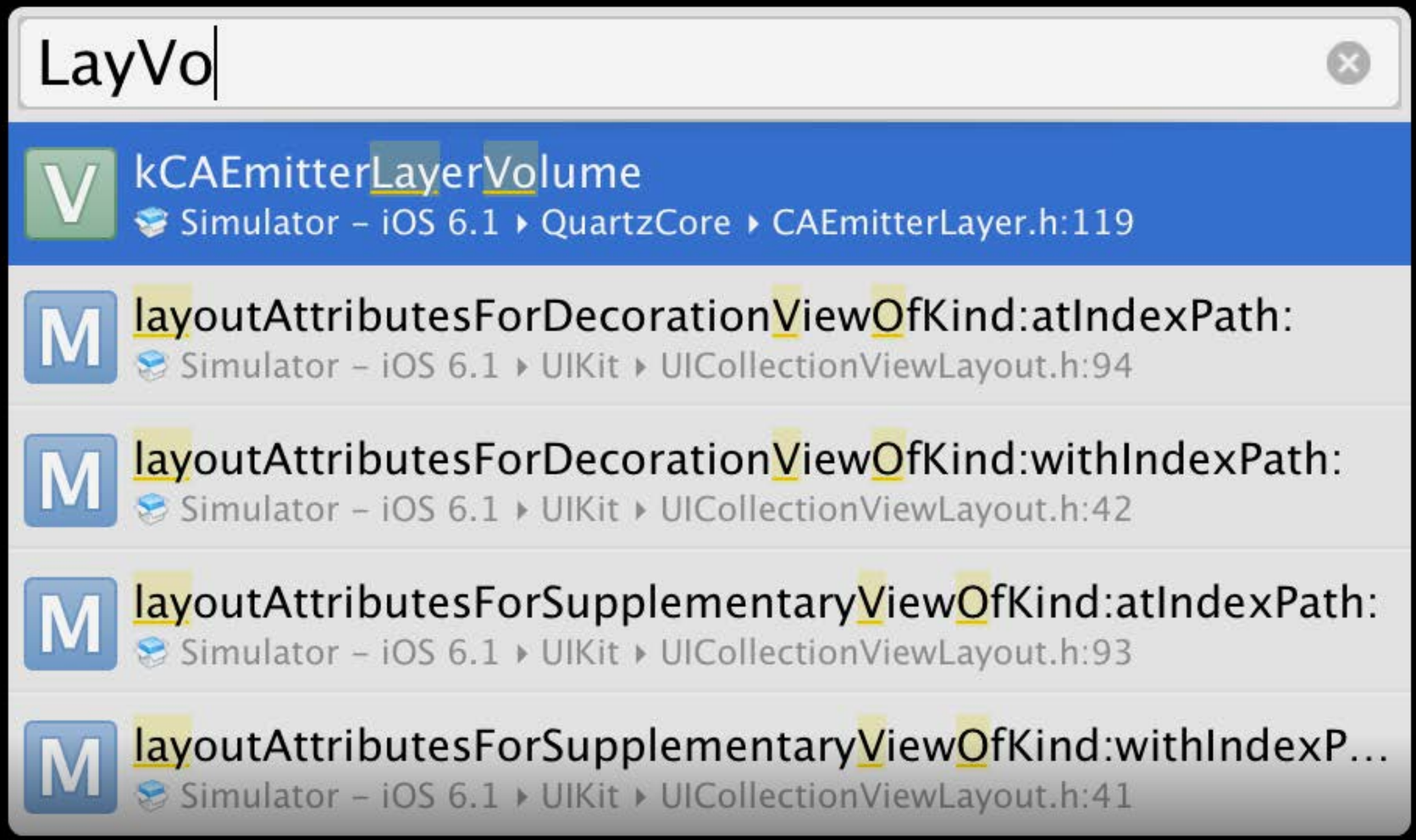
Open Quickly



Open Quickly



Open Quickly



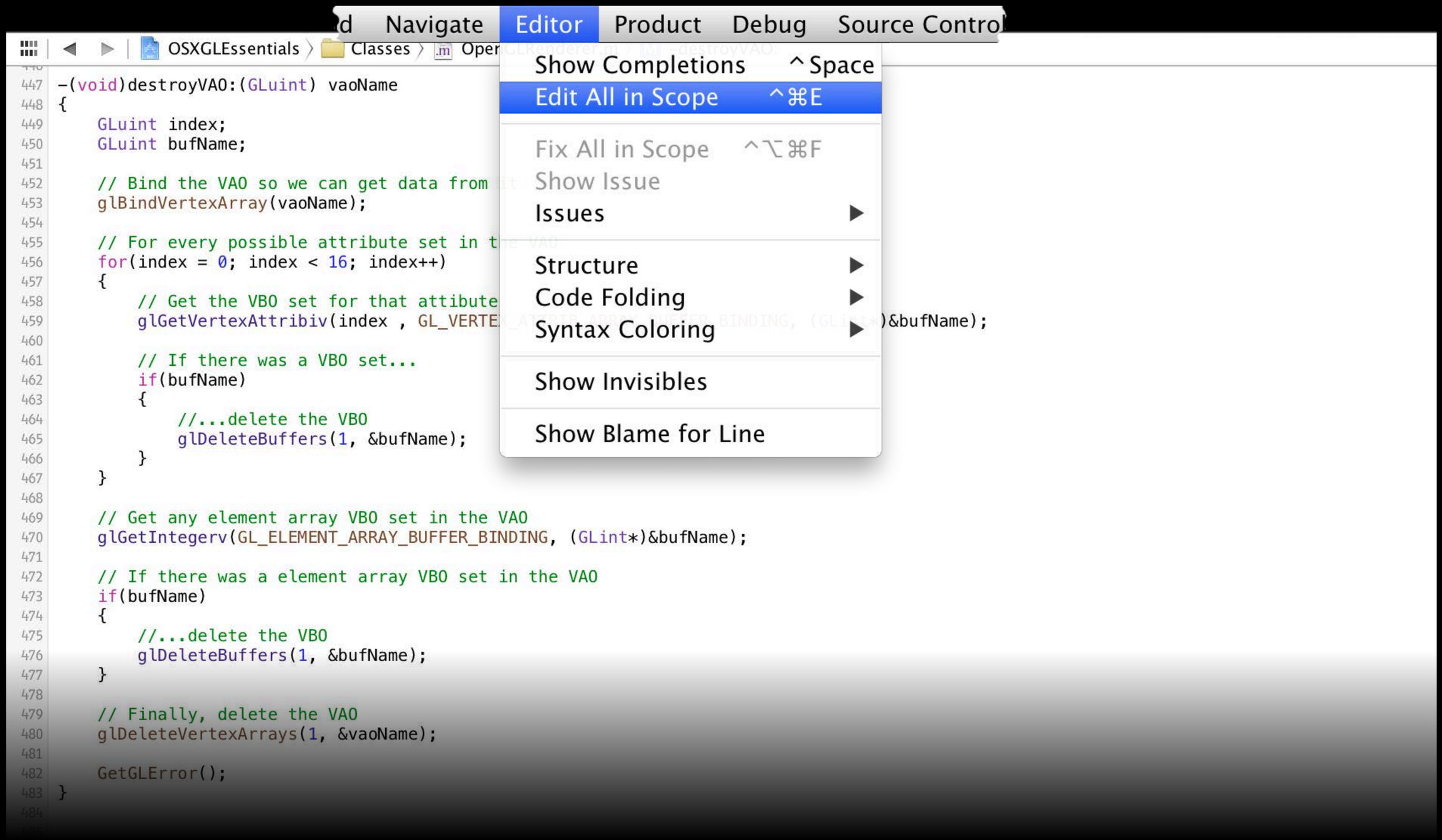
Source Editor Power Tricks

Edit All in Scope

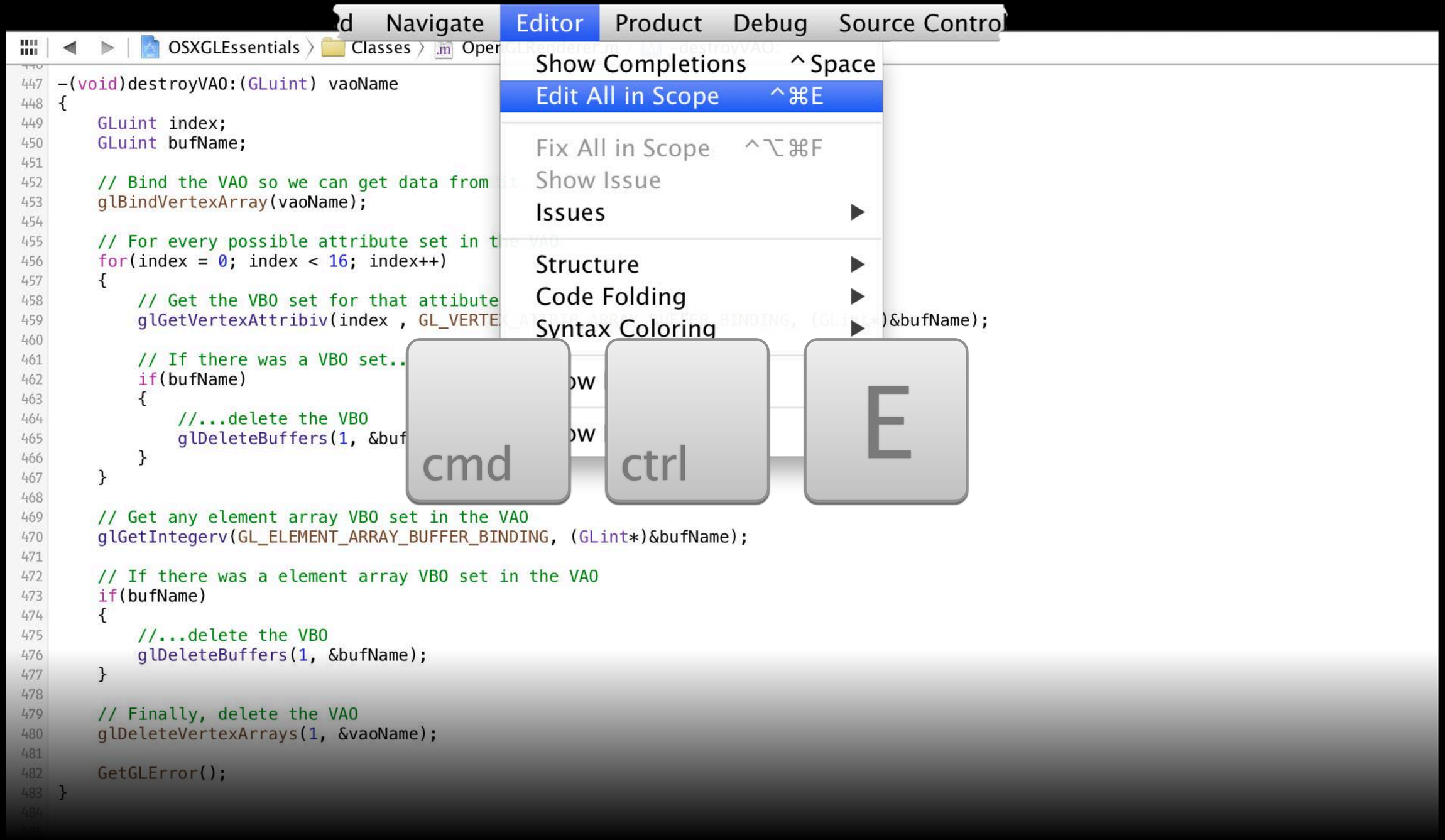
OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
447 -(void)destroyVAO:(GLuint) vaoName
448 {
449     GLuint index;
450     GLuint bufName;
451
452     // Bind the VAO so we can get data from it
453     glBindVertexArray(vaoName);
454
455     // For every possible attribute set in the VAO
456     for(index = 0; index < 16; index++)
457     {
458         // Get the VBO set for that attibute
459         glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufName);
460
461         // If there was a VBO set...
462         if(bufName)
463         {
464             //...delete the VBO
465             glDeleteBuffers(1, &bufName);
466         }
467     }
468
469     // Get any element array VBO set in the VAO
470     glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufName);
471
472     // If there was a element array VBO set in the VAO
473     if(bufName)
474     {
475         //...delete the VBO
476         glDeleteBuffers(1, &bufName);
477     }
478
479     // Finally, delete the VAO
480     glDeleteVertexArrays(1, &vaoName);
481
482     GetGLError();
483 }
484
```

Edit All in Scope



Edit All in Scope



Edit All in Scope

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

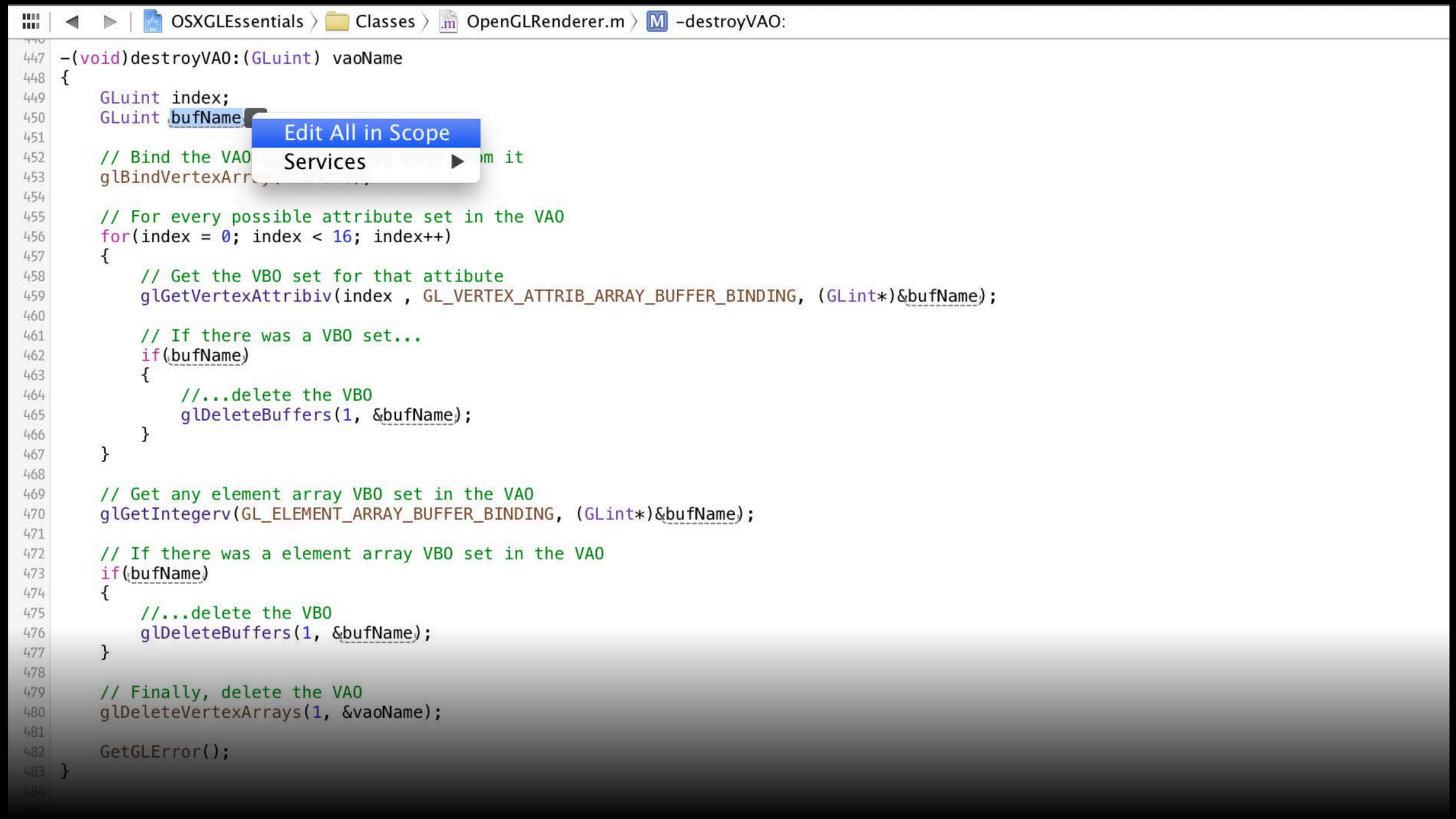
```
447 -(void)destroyVAO:(GLuint) vaoName
448 {
449     GLuint index;
450     GLuint bufName;
451
452     // Bind the VAO so we can get data from it
453     glBindVertexArray(vaoName);
454
455     // For every possible attribute set in the VAO
456     for(index = 0; index < 16; index++)
457     {
458         // Get the VBO set for that attibute
459         glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufName);
460
461         // If there was a VBO set...
462         if(bufName)
463         {
464             //...delete the VBO
465             glDeleteBuffers(1, &bufName);
466         }
467     }
468
469     // Get any element array VBO set in the VAO
470     glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufName);
471
472     // If there was a element array VBO set in the VAO
473     if(bufName)
474     {
475         //...delete the VBO
476         glDeleteBuffers(1, &bufName);
477     }
478
479     // Finally, delete the VAO
480     glDeleteVertexArrays(1, &vaoName);
481
482     GetGLError();
483 }
484
```

Edit All in Scope

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
-(void)destroyVAO:(GLuint) vaoName
{
    GLuint index;
    GLuint bufName;

    // Bind the VAO so we can get data from it
    glBindVertexArray(vaoName);

    // For every possible attribute set in the VAO
    for(index = 0; index < 16; index++)
    {
        // Get the VBO set for that attibute
        glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufName);

        // If there was a VBO set...
        if(bufName)
        {
            //...delete the VBO
            glDeleteBuffers(1, &bufName);
        }
    }

    // Get any element array VBO set in the VAO
    glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufName);

    // If there was a element array VBO set in the VAO
    if(bufName)
    {
        //...delete the VBO
        glDeleteBuffers(1, &bufName);
    }

    // Finally, delete the VAO
    glDeleteVertexArrays(1, &vaoName);

    GetGLError();
}
```

Edit All in Scope



Edit All in Scope

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
447 -(void)destroyVAO:(GLuint) vaoName
448 {
449     GLuint index;
450     GLuint bufName;
451
452     // Bind the VAO so we can get data from it
453     glBindVertexArray(vaoName);
454
455     // For every possible attribute set in the VAO
456     for(index = 0; index < 16; index++)
457     {
458         // Get the VBO set for that attibute
459         glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufName);
460
461         // If there was a VBO set...
462         if(bufName)
463         {
464             //...delete the VBO
465             glDeleteBuffers(1, &bufName);
466         }
467     }
468
469     // Get any element array VBO set in the VAO
470     glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufName);
471
472     // If there was a element array VBO set in the VAO
473     if(bufName)
474     {
475         //...delete the VBO
476         glDeleteBuffers(1, &bufName);
477     }
478
479     // Finally, delete the VAO
480     glDeleteVertexArrays(1, &vaoName);
481
482     GetGLError();
483 }
484
```

Edit All in Scope

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
447 -(void)destroyVAO:(GLuint) vaoName
448 {
449     GLuint index;
450     GLuint b;
451
452     // Bind the VAO so we can get data from it
453     glBindVertexArray(vaoName);
454
455     // For every possible attribute set in the VAO
456     for(index = 0; index < 16; index++)
457     {
458         // Get the VBO set for that attibute
459         glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&b);
460
461         // If there was a VBO set...
462         if(b)
463         {
464             //...delete the VBO
465             glDeleteBuffers(1, &b);
466         }
467     }
468
469     // Get any element array VBO set in the VAO
470     glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&b);
471
472     // If there was a element array VBO set in the VAO
473     if(b)
474     {
475         //...delete the VBO
476         glDeleteBuffers(1, &b);
477     }
478
479     // Finally, delete the VAO
480     glDeleteVertexArrays(1, &vaoName);
481
482     GetGLError();
483 }
484
```

Edit All in Scope

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
447 -(void)destroyVAO:(GLuint) vaoName
448 {
449     GLuint index;
450     GLuint bu;
451
452     // Bind the VAO so we can get data from it
453     glBindVertexArray(vaoName);
454
455     // For every possible attribute set in the VAO
456     for(index = 0; index < 16; index++)
457     {
458         // Get the VBO set for that attibute
459         glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bu);
460
461         // If there was a VBO set...
462         if(bu)
463         {
464             //...delete the VBO
465             glDeleteBuffers(1, &bu);
466         }
467     }
468
469     // Get any element array VBO set in the VAO
470     glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bu);
471
472     // If there was a element array VBO set in the VAO
473     if(bu)
474     {
475         //...delete the VBO
476         glDeleteBuffers(1, &bu);
477     }
478
479     // Finally, delete the VAO
480     glDeleteVertexArrays(1, &vaoName);
481
482     GetGLError();
483 }
484
```

Edit All in Scope

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
447 -(void)destroyVAO:(GLuint) vaoName
448 {
449     GLuint index;
450     GLuint buf;
451
452     // Bind the VAO so we can get data from it
453     glBindVertexArray(vaoName);
454
455     // For every possible attribute set in the VAO
456     for(index = 0; index < 16; index++)
457     {
458         // Get the VBO set for that attibute
459         glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&buf);
460
461         // If there was a VBO set...
462         if(buf)
463         {
464             //...delete the VBO
465             glDeleteBuffers(1, &buf);
466         }
467     }
468
469     // Get any element array VBO set in the VAO
470     glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&buf);
471
472     // If there was a element array VBO set in the VAO
473     if(buf)
474     {
475         //...delete the VBO
476         glDeleteBuffers(1, &buf);
477     }
478
479     // Finally, delete the VAO
480     glDeleteVertexArrays(1, &vaoName);
481
482     GetGLError();
483 }
484
```

Edit All in Scope

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
447 -(void)destroyVAO:(GLuint) vaoName
448 {
449     GLuint index;
450     GLuint bufI;
451
452     // Bind the VAO so we can get data from it
453     glBindVertexArray(vaoName);
454
455     // For every possible attribute set in the VAO
456     for(index = 0; index < 16; index++)
457     {
458         // Get the VBO set for that attibute
459         glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufI);
460
461         // If there was a VBO set...
462         if(bufI)
463         {
464             //...delete the VBO
465             glDeleteBuffers(1, &bufI);
466         }
467     }
468
469     // Get any element array VBO set in the VAO
470     glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufI);
471
472     // If there was a element array VBO set in the VAO
473     if(bufI)
474     {
475         //...delete the VBO
476         glDeleteBuffers(1, &bufI);
477     }
478
479     // Finally, delete the VAO
480     glDeleteVertexArrays(1, &vaoName);
481
482     GetGLError();
483 }
484
```

Edit All in Scope

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
447 -(void)destroyVAO:(GLuint) vaoName
448 {
449     GLuint index;
450     GLuint bufId;
451
452     // Bind the VAO so we can get data from it
453     glBindVertexArray(vaoName);
454
455     // For every possible attribute set in the VAO
456     for(index = 0; index < 16; index++)
457     {
458         // Get the VBO set for that attibute
459         glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufId);
460
461         // If there was a VBO set...
462         if(bufId)
463         {
464             //...delete the VBO
465             glDeleteBuffers(1, &bufId);
466         }
467     }
468
469     // Get any element array VBO set in the VAO
470     glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufId);
471
472     // If there was a element array VBO set in the VAO
473     if(bufId)
474     {
475         //...delete the VBO
476         glDeleteBuffers(1, &bufId);
477     }
478
479     // Finally, delete the VAO
480     glDeleteVertexArrays(1, &vaoName);
481
482     GetGLError();
483 }
484
```

Edit All in Scope

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
447 -(void)destroyVAO:(GLuint) vaoName
448 {
449     GLuint index;
450     GLuint bufIde;
451
452     // Bind the VAO so we can get data from it
453     glBindVertexArray(vaoName);
454
455     // For every possible attribute set in the VAO
456     for(index = 0; index < 16; index++)
457     {
458         // Get the VBO set for that attibute
459         glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufIde);
460
461         // If there was a VBO set...
462         if(bufIde)
463         {
464             //...delete the VBO
465             glDeleteBuffers(1, &bufIde);
466         }
467     }
468
469     // Get any element array VBO set in the VAO
470     glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufIde);
471
472     // If there was a element array VBO set in the VAO
473     if(bufIde)
474     {
475         //...delete the VBO
476         glDeleteBuffers(1, &bufIde);
477     }
478
479     // Finally, delete the VAO
480     glDeleteVertexArrays(1, &vaoName);
481
482     GetGLError();
483 }
484
```

Edit All in Scope

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
447 -(void)destroyVAO:(GLuint) vaoName
448 {
449     GLuint index;
450     GLuint bufIden;
451
452     // Bind the VAO so we can get data from it
453     glBindVertexArray(vaoName);
454
455     // For every possible attribute set in the VAO
456     for(index = 0; index < 16; index++)
457     {
458         // Get the VBO set for that attibute
459         glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufIden);
460
461         // If there was a VBO set...
462         if(bufIden)
463         {
464             //...delete the VBO
465             glDeleteBuffers(1, &bufIden);
466         }
467     }
468
469     // Get any element array VBO set in the VAO
470     glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufIden);
471
472     // If there was a element array VBO set in the VAO
473     if(bufIden)
474     {
475         //...delete the VBO
476         glDeleteBuffers(1, &bufIden);
477     }
478
479     // Finally, delete the VAO
480     glDeleteVertexArrays(1, &vaoName);
481
482     GetGLError();
483 }
484
```

Edit All in Scope

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
447 -(void)destroyVAO:(GLuint) vaoName
448 {
449     GLuint index;
450     GLuint bufIdent;
451
452     // Bind the VAO so we can get data from it
453     glBindVertexArray(vaoName);
454
455     // For every possible attribute set in the VAO
456     for(index = 0; index < 16; index++)
457     {
458         // Get the VBO set for that attibute
459         glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);
460
461         // If there was a VBO set...
462         if(bufIdent)
463         {
464             //...delete the VBO
465             glDeleteBuffers(1, &bufIdent);
466         }
467     }
468
469     // Get any element array VBO set in the VAO
470     glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);
471
472     // If there was a element array VBO set in the VAO
473     if(bufIdent)
474     {
475         //...delete the VBO
476         glDeleteBuffers(1, &bufIdent);
477     }
478
479     // Finally, delete the VAO
480     glDeleteVertexArrays(1, &vaoName);
481
482     GetGLError();
483 }
484
```

Edit All in Scope

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
447 -(void)destroyVAO:(GLuint) vaoName
448 {
449     GLuint index;
450     GLuint bufIdent;
451
452     // Bind the VAO so we can get data from it
453     glBindVertexArray(vaoName);
454
455     // For every possible attribute set in the VAO
456     for(index = 0; index < 16; index++)
457     {
458         // Get the VBO set for that attibute
459         glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);
460
461         // If there was a VBO set...
462         if(bufIdent)
463         {
464             //...delete the VBO
465             glDeleteBuffers(1, &bufIdent);
466         }
467     }
468
469     // Get any element array VBO set in the VAO
470     glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);
471
472     // If there was a element array VBO set in the VAO
473     if(bufIdent)
474     {
475         //...delete the VBO
476         glDeleteBuffers(1, &bufIdent);
477     }
478
479     // Finally, delete the VAO
480     glDeleteVertexArrays(1, &vaoName);
481
482     GetGLError();
483 }
484
```

Code Snippets

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
-(void)destroyVAO:(GLuint) vaoName
{
    GLuint index;
    GLuint bufIdent;

    // Bind the VAO so we can get data from it
    glBindVertexArray(vaoName);



    // For every possible attribute set in the VAO
    for(index = 0; index < 16; index++)
    {
        // Get the VBO set for that attibute
        glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);

        // If there was a VBO set...
        if(bufIdent)
        {
            //...delete the VBO
            glDeleteBuffers(1, &bufIdent);
        }
    }

    // Get any element array VBO set in the VAO
    glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);

    // If there was a element array VBO set in the VAO
    if(bufIdent)
    {
        //...delete the VBO
        glDeleteBuffers(1, &bufIdent);
    }

    // Finally, delete the VAO
    glDeleteVertexArrays(1, &vaoName);

    GetGLError();
}
```

Identity and Type

Name: OpenGLRenderer.m

Type: Default – Objective-C So…

Location: Relative to Group

Classes/
OpenGLRenderer.m

Full Path: /Users/anders/Desktop/
OpenGLEssentials/Classes/
OpenGLRenderer.m

C Block typedef – Used for defining a block as a type.

C Inline Block as Variable – Used for saving a block to a variable so we can pass it as an argument multiple…

C typedef – Used for defining a type.

C++ Class Declaration – Used for describing a new class type containing instance variables,…

C++ Class Template – Used to define a new class template.

Code Snippets

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
-(void)destroyVAO:(GLuint) vaoName
{
    GLuint index;
    GLuint bufIdent;

    // Bind the VAO so we can get data from it
    glBindVertexArray(vaoName);

    Return Type(^Block Name)(Parameter Types) = ^(Parameters) {
        Code
    };

    // For every possible attribute set in the VAO
    for(index = 0; index < 16; index++)
    {
        // Get the VBO set for that attibute
        glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);

        // If there was a VBO set...
        if(bufIdent)
        {
            //...delete the VBO
            glDeleteBuffers(1, &bufIdent);
        }
    }

    // Get any element array VBO set in the VAO
    glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);

    // If there was a element array VBO set in the VAO
    if(bufIdent)
    {
        //...delete the VBO
        glDeleteBuffers(1, &bufIdent);
    }

    // Finally, delete the VAO
    glDeleteVertexArrays(1, &vaoName);
```

Identity and Type

Name: OpenGLRenderer.m

Type: Default – Objective-C So…

Location: Relative to Group

Classes/
OpenGLRenderer.m

Full Path: /Users/anders/Desktop/
OpenGLEssentials/Classes/
OpenGLRenderer.m

C Block typedef – Used for defining a block as a type.

C Inline Block as Variable – Used for saving a block to a variable so we can pass it as an argument multiple…

C typedef – Used for defining a type.

C++ Class Declaration – Used for describing a new class type containing instance variables,…

C++ Class Template – Used to define a new class template.

Code Snippets

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
-(void)destroyVAO:(GLuint) vaoName
{
    GLuint index;
    GLuint bufIdent;

    // Bind the VAO so we can get data from it
    glBindVertexArray(vaoName);

    Return Type(^Block Name)(Parameter Types) = ^(Parameters) {
        Code
    };

    // For every possible attribute set in the VAO
    for(index = 0; index < 16; index++)
    {
        // Get the VBO set for that attibute
        glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);

        // If there was a VBO set...
        if(bufIdent)
        {
            //...delete the VBO
            glDeleteBuffers(1, &bufIdent);
        }
    }

    // Get any element array VBO set in the VAO
    glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);

    // If there was a element array VBO set in the VAO
    if(bufIdent)
    {
        //...delete the VBO
        glDeleteBuffers(1, &bufIdent);
    }

    // Finally, delete the VAO
    glDeleteVertexArrays(1, &vaoName);
```

Identity and Type

Name: OpenGLRenderer.m

Type: Default – Objective-C So…

Location: Relative to Group

Classes/
OpenGLRenderer.m

Full Path: /Users/anders/Desktop/
OpenGLEssentials/Classes/
OpenGLRenderer.m

C Block typedef – Used for defining a block as a type.

C Inline Block as Variable – Used for saving a block to a variable so we can pass it as an argument multiple…

C typedef – Used for defining a type.

C++ Class Declaration – Used for describing a new class type containing instance variables,…

C++ Class Template – Used to define a new class template.

Code Snippets

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
-(void)destroyVAO:(GLuint) vaoName
{
    GLuint index;
    GLuint bufIdent;

    // Bind the VAO so we can get data from it
    glBindVertexArray(vaoName);

    void (^deleteBuffer)(GLint) = ^(GLint ident) {
        glDeleteBuffers(1, &ident);
    };

    // For every possible attribute set in the VAO
    for(index = 0; index < 16; index++)
    {
        // Get the VBO set for that attibute
        glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);

        // If there was a VBO set...
        if(bufIdent)
        {
            //...delete the VBO
            glDeleteBuffers(1, &bufIdent);
        }
    }

    // Get any element array VBO set in the VAO
    glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);

    // If there was a element array VBO set in the VAO
    if(bufIdent)
    {
        //...delete the VBO
        glDeleteBuffers(1, &bufIdent);
    }

    // Finally, delete the VAO
    glDeleteVertexArrays(1, &vaoName);
```

Identity and Type

Name: OpenGLRenderer.m

Type: Default – Objective-C So…

Location: Relative to Group

Classes/
OpenGLRenderer.m

Full Path: /Users/anders/Desktop/
OpenGLEssentials/Classes/
OpenGLRenderer.m

C Block typedef – Used for defining a block as a type.

C Inline Block as Variable – Used for saving a block to a variable so we can pass it as an argument multiple…

C typedef – Used for defining a type.

C++ Class Declaration – Used for describing a new class type containing instance variables,…

C++ Class Template – Used to define a new class template.

Custom Code Snippets

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
447 -(void)destroyVAO:(GLuint) vaoName
448 {
449     GLuint index;
450     GLuint bufIdent;
451
452     // Bind the VAO so we can get data from it
453     glBindVertexArray(vaoName);
454
455     void (^deleteBuffer)(GLint) = ^(GLint ident) {
456         glDeleteBuffers(1, &ident);
457     };
458
459     // For every possible attribute set in the VAO
460     for(index = 0; index < 16; index++)
461     {
462         // Get the VBO set for that attibute
463         glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);
464
465         // If there was a VBO set...
466         if(bufIdent)
467         {
468             //...delete the VBO
469             glDeleteBuffers(1, &bufIdent);
470         }
471     }
472
473     // Get any element array VBO set in the VAO
474     glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);
475
476     // If there was a element array VBO set in the VAO
477     if(bufIdent)
478     {
479         //...delete the VBO
480         glDeleteBuffers(1, &bufIdent);
481     }
482
483     // Finally, delete the VAO
484     glDeleteVertexArrays(1, &vaoName);
```

Identity and Type

Name: OpenGLRenderer.m

Type: Default – Objective-C So…

Location: Relative to Group

Classes/
OpenGLRenderer.m

Full Path: /Users/anders/Desktop/
OpenGLEssentials/Classes/
OpenGLRenderer.m

C Block typedef – Used for defining a block as a type.

C Inline Block as Variable – Used for saving a block to a variable so we can pass it as an argument multiple…

C typedef – Used for defining a type.

C++ Class Declaration – Used for describing a new class type containing instance variables,…

C++ Class Template – Used to define a new class template.

Custom Code Snippets

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
447 -(void)destroyVAO:(GLuint) vaoName
448 {
449     GLuint index;
450     GLuint bufIdent;
451
452     // Bind the VAO so we can get data from it
453     glBindVertexArray(vaoName);
454
455     void (^deleteBuffer)(GLint) = ^(GLint ident) {
456         glDeleteBuffers(1, &ident);
457     };
458
459     // For every possible attribute set in the VAO
460     for(index = 0; index < 16; index++)
461     {
462         // Get the VBO set for that attibute
463         glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);
464
465         // If there was a VBO set...
466         if(bufIdent)
467         {
468             //...delete the VBO
469             glDeleteBuffers(1, &bufIdent);
470         }
471     }
472
473     // Get any element array VBO set in the VAO
474     glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);
475
476     // If there was a element array VBO set in the VAO
477     if(bufIdent)
478     {
479         //...delete the VBO
480         glDeleteBuffers(1, &bufIdent);
481     }
482
483     // Finally, delete the VAO
484     glDeleteVertexArrays(1, &vaoName);
```

Identity and Type

Name: OpenGLRenderer.m

Type: Default – Objective-C So…

Location: Relative to Group

Classes/
OpenGLRenderer.m

Full Path: /Users/anders/Desktop/
OpenGLEssentials/Classes/
OpenGLRenderer.m

C Block typedef – Used for defining a block as a type.

C Inline Block as Variable – Used for saving a block to a variable so we can pass it as an argument multiple…

C typedef – Used for defining a type.

C++ Class Declaration – Used for describing a new class type containing instance variables,…

C++ Class Template – Used to define a new class template.

Custom Code Snippets

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
447 -(void)destroyVAO:(GLuint) vaoName
448 {
449     GLuint index;
450     GLuint bufIdent;
451
452     // Bind the VAO so we can get data from it
453     glBindVertexArray(vaoName);
454
455     void (^deleteBuffer)(GLint) = ^(GLint ident) {
456         glDeleteBuffers(1, &ident);
457     };
458
459     // For every possible attribute set in the VAO
460     for(index = 0; index < 16; index++)
461     {
462         // Get the VBO set for that attibute
463         glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);
464
465         // If there was a VBO set...
466         if(bufIdent)
467         {
468             //...delete the VBO
469             glDeleteBuffers(1, &bufIdent);
470         }
471     }
472
473     // Get any element array VBO set in the VAO
474     glGetIntegerv(GL_ELEMENT_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);
475
476     // If there was a element array VBO set in the VAO
477     if(bufIdent)
478     {
479         //...delete the VBO
480         glDeleteBuffers(1, &bufIdent);
481     }
482
483     // Finally, delete the VAO
484     glDeleteVertexArrays(1, &vaoName);
```

Identity and Type

Name: OpenGLRenderer.m

Type: Default – Objective-C So…

Location: Relative to Group

Classes/
OpenGLRenderer.m

Full Path: /Users/anders/Desktop/
OpenGLEssentials/Classes/
OpenGLRenderer.m

My Code Snippet (User)

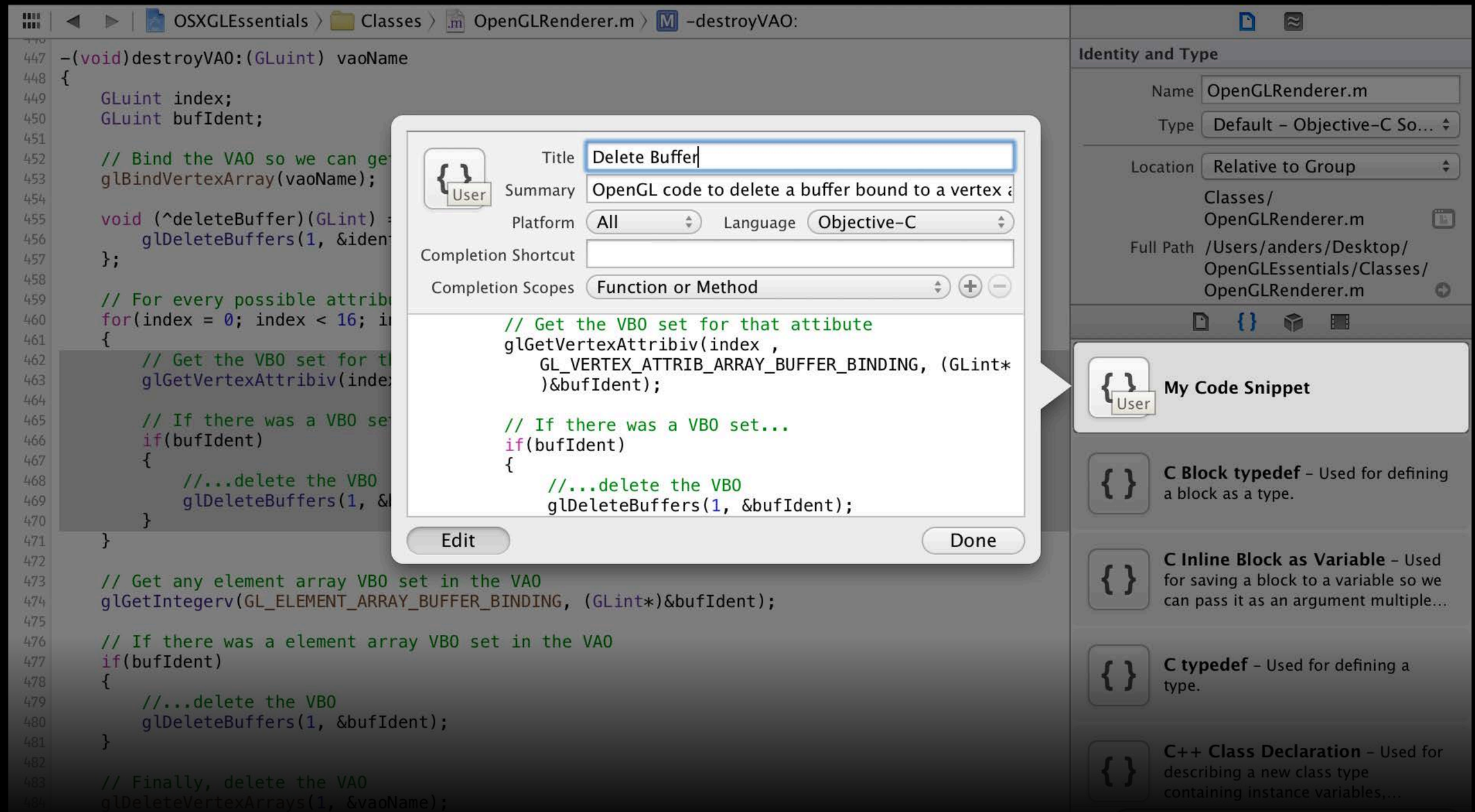
C Block typedef – Used for defining a block as a type.

C Inline Block as Variable – Used for saving a block to a variable so we can pass it as an argument multiple…

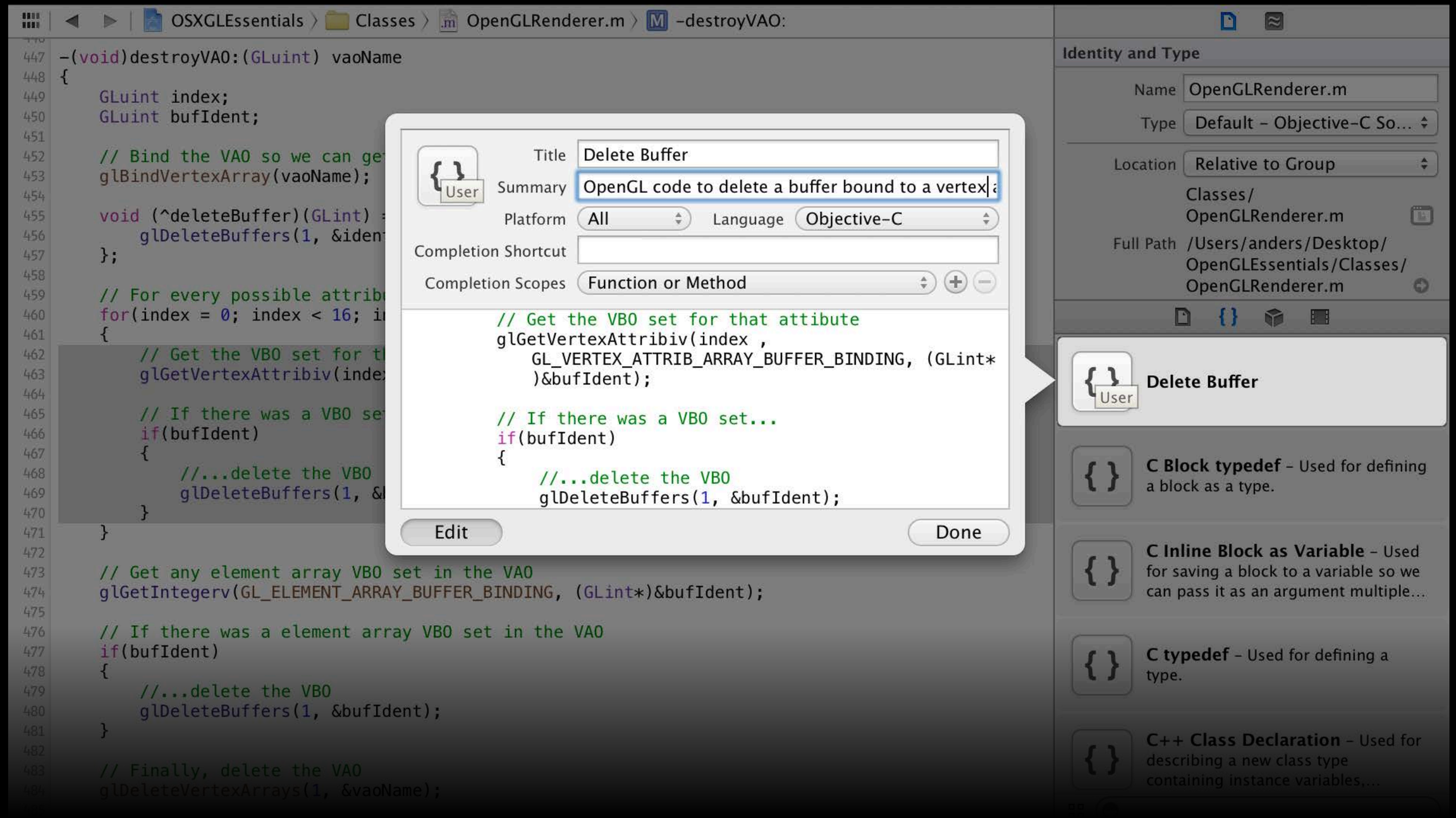
C typedef – Used for defining a type.

C++ Class Declaration – Used for describing a new class type containing instance variables,…

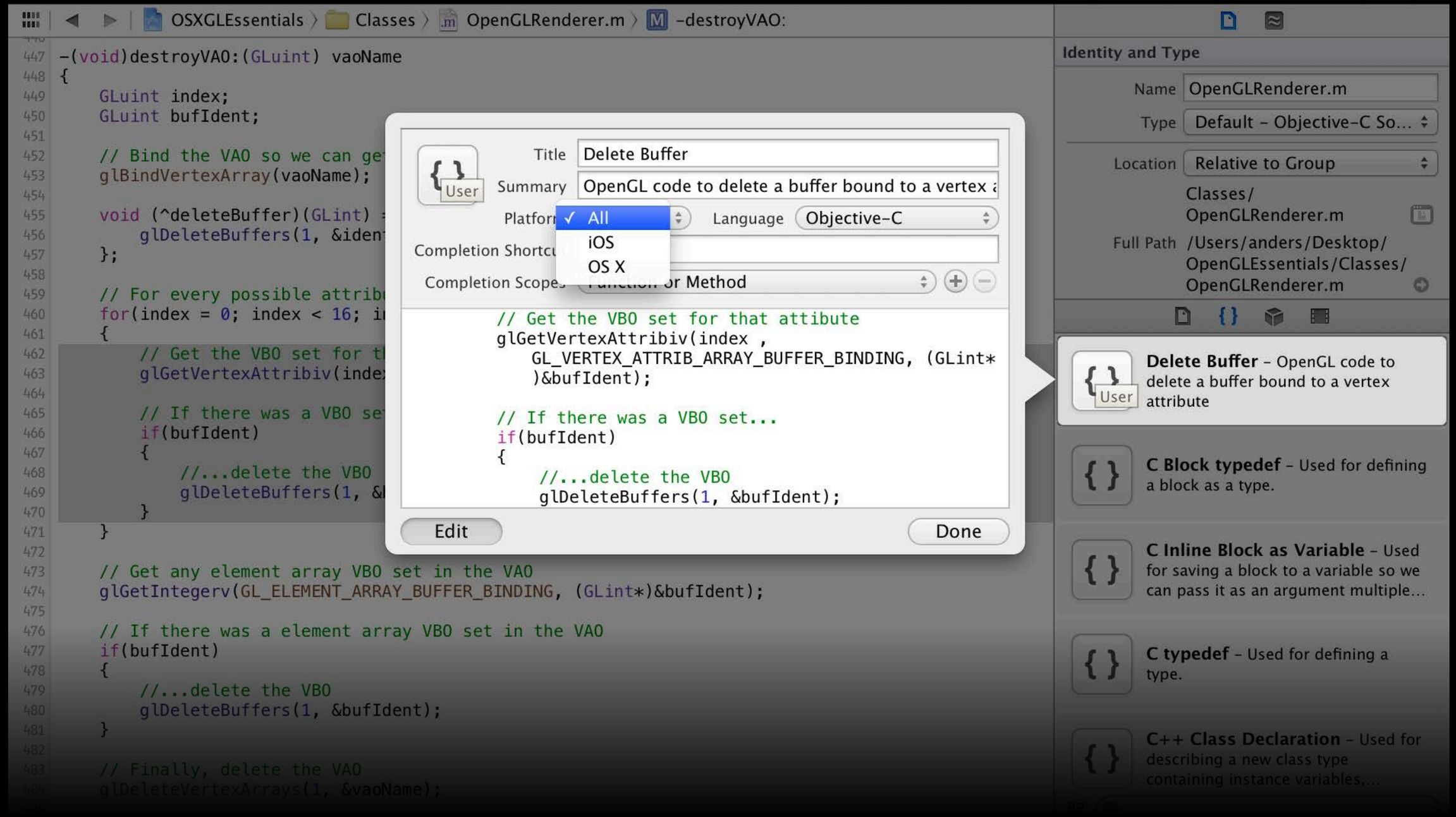
Custom Code Snippets



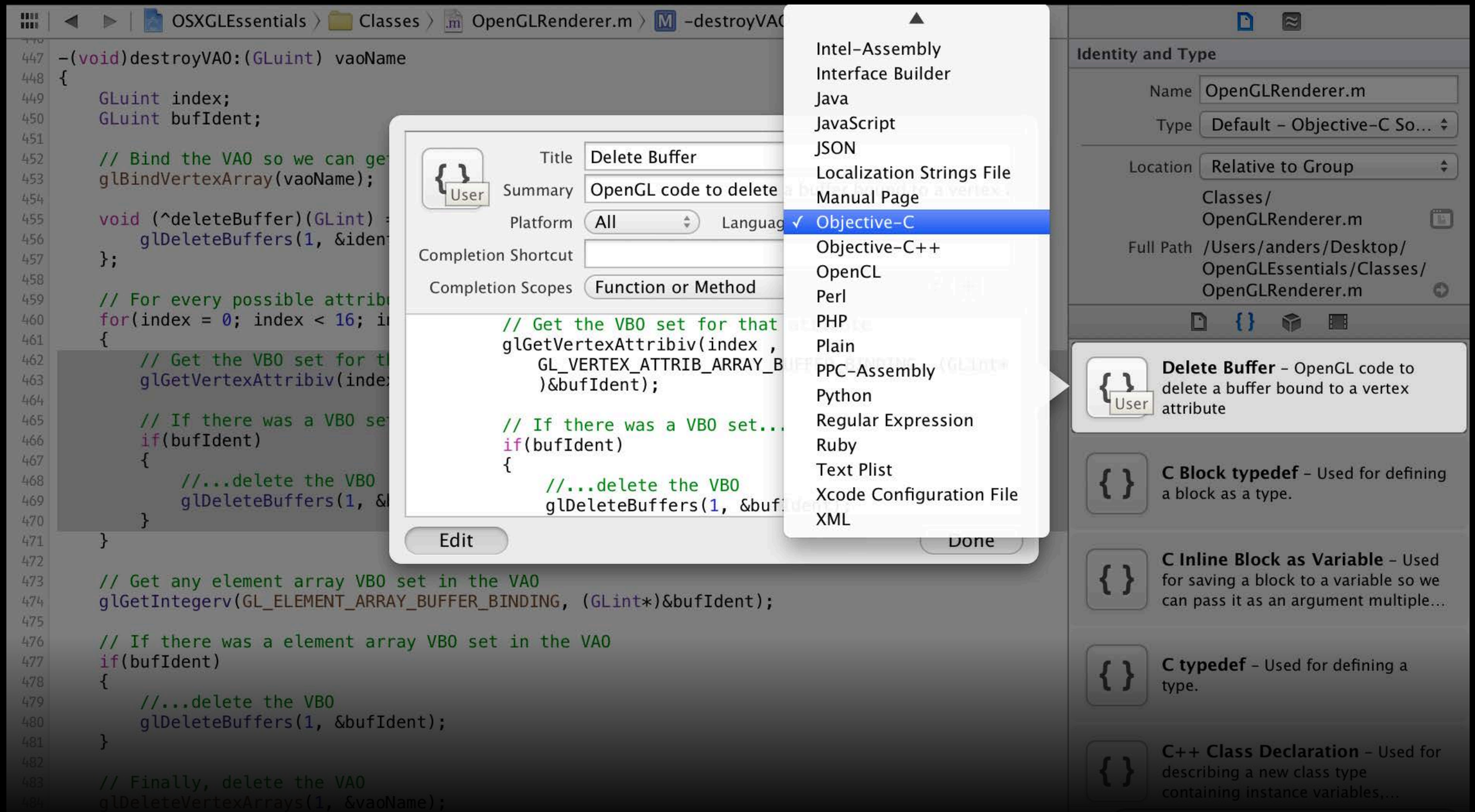
Custom Code Snippets



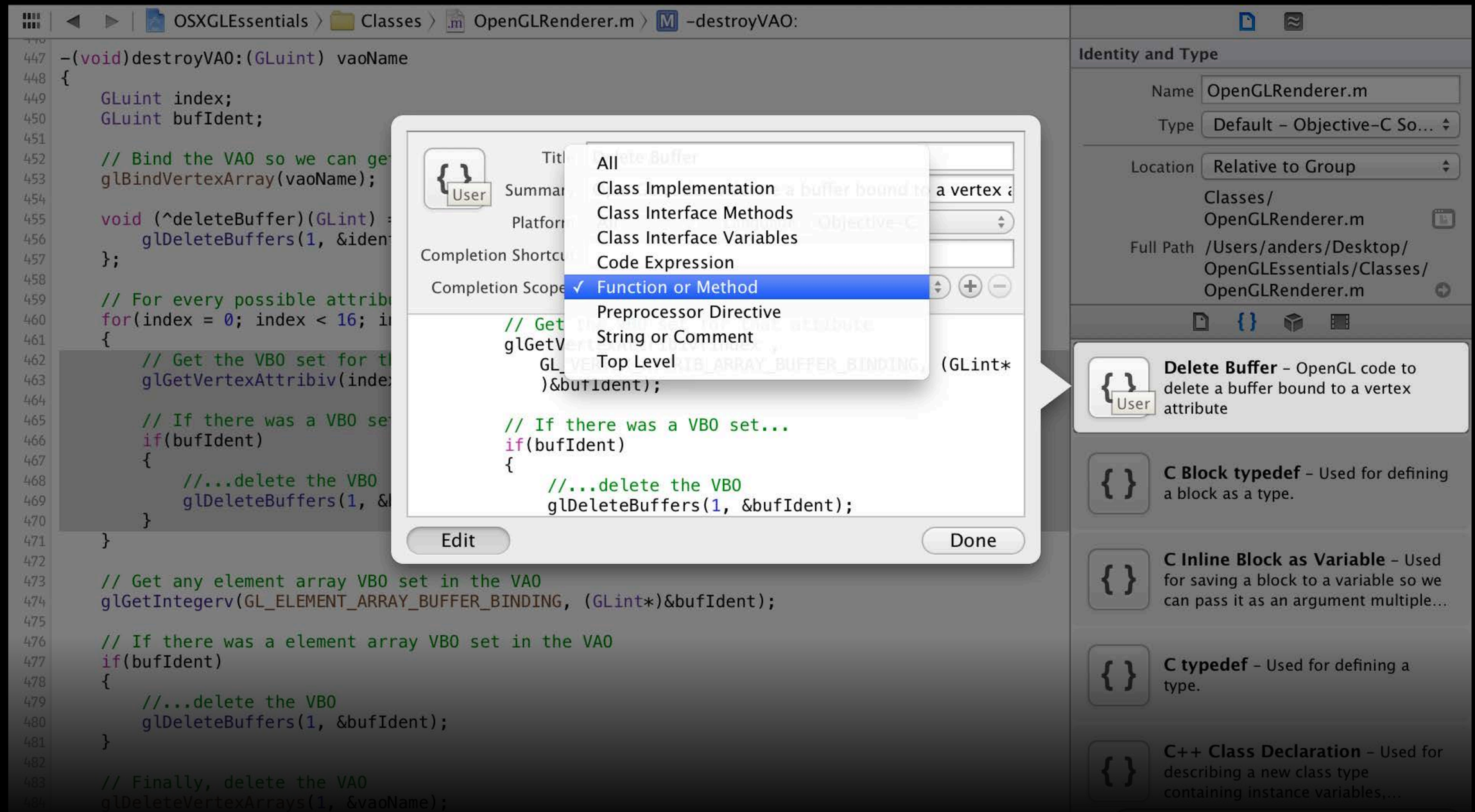
Custom Code Snippets



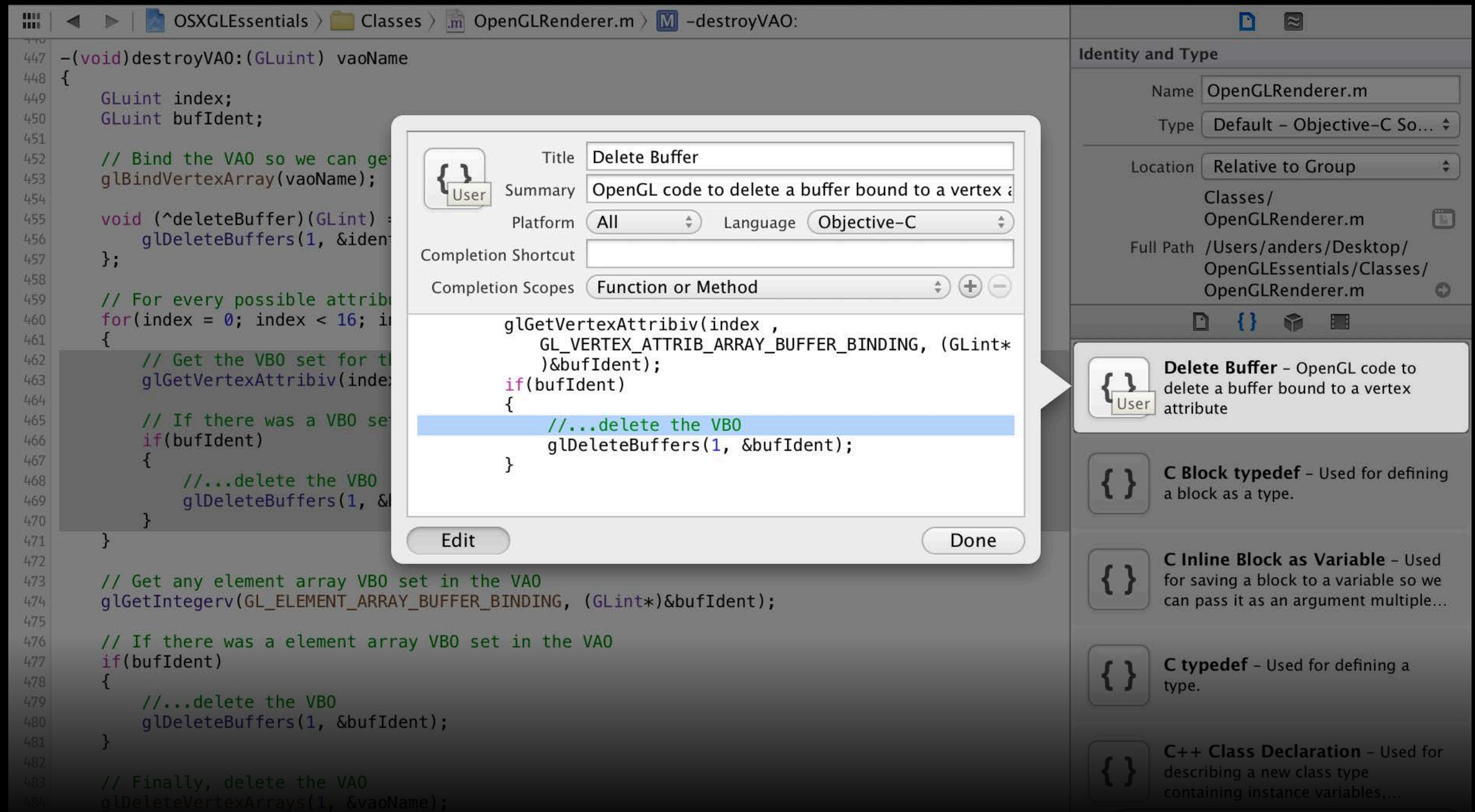
Custom Code Snippets



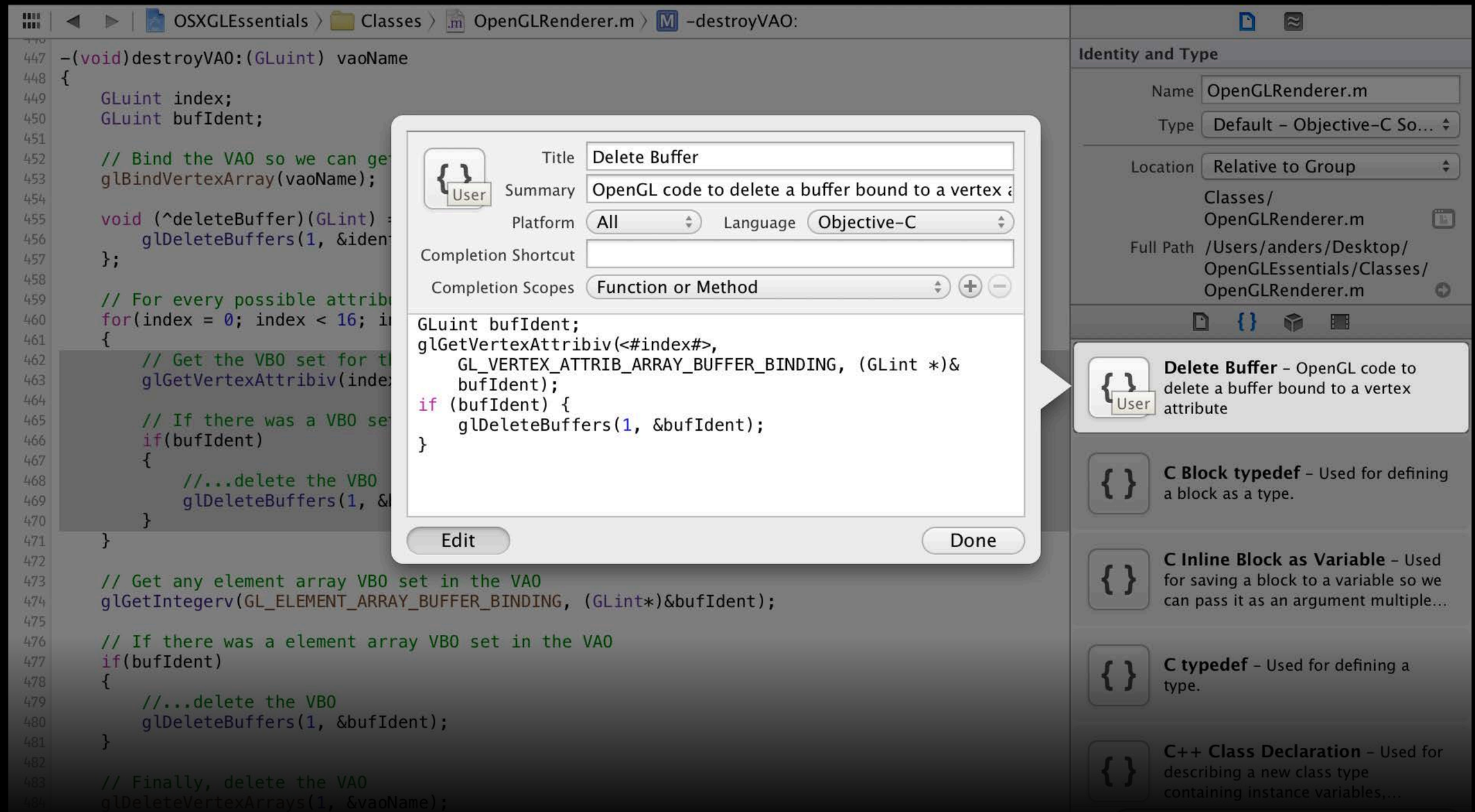
Custom Code Snippets



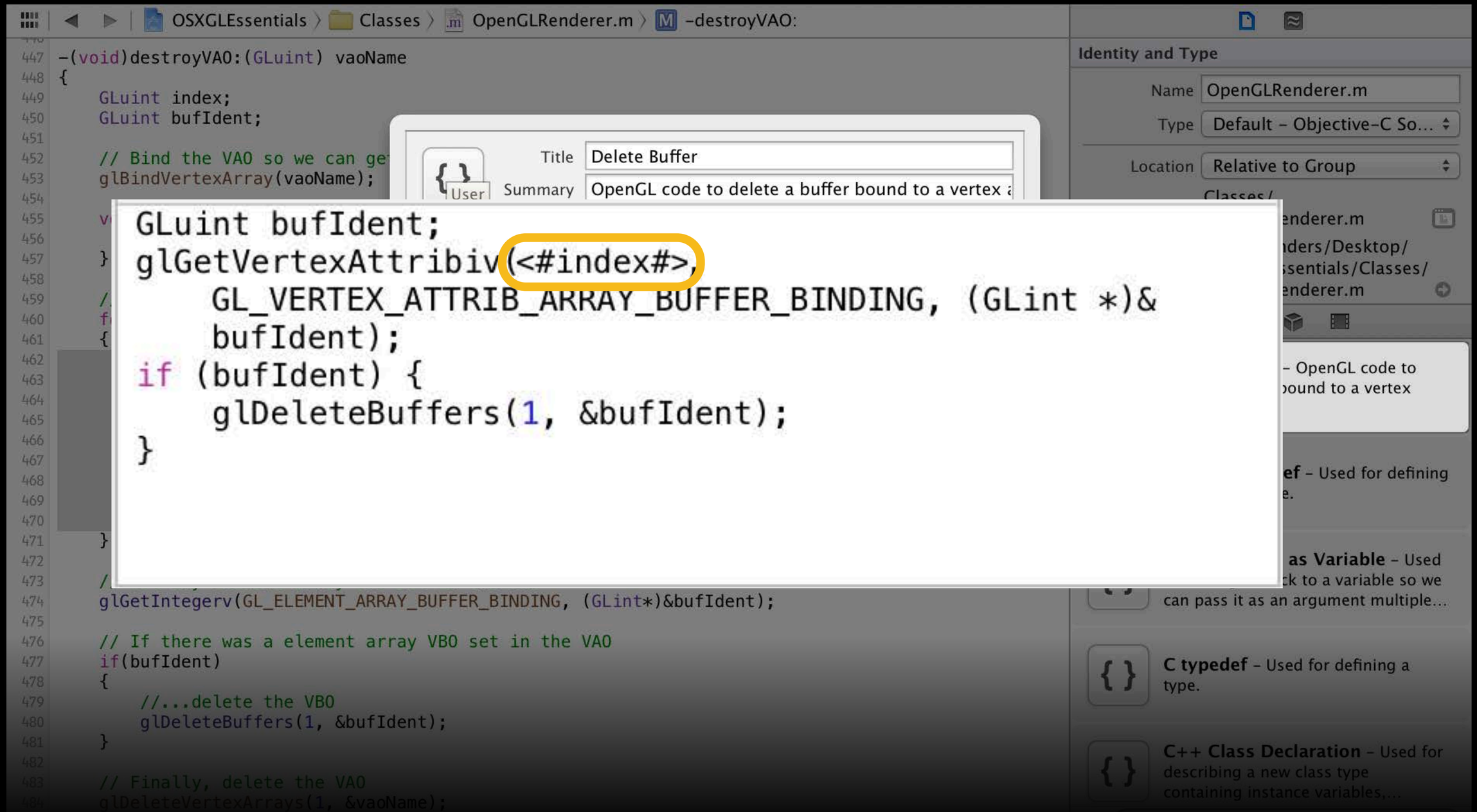
Custom Code Snippets



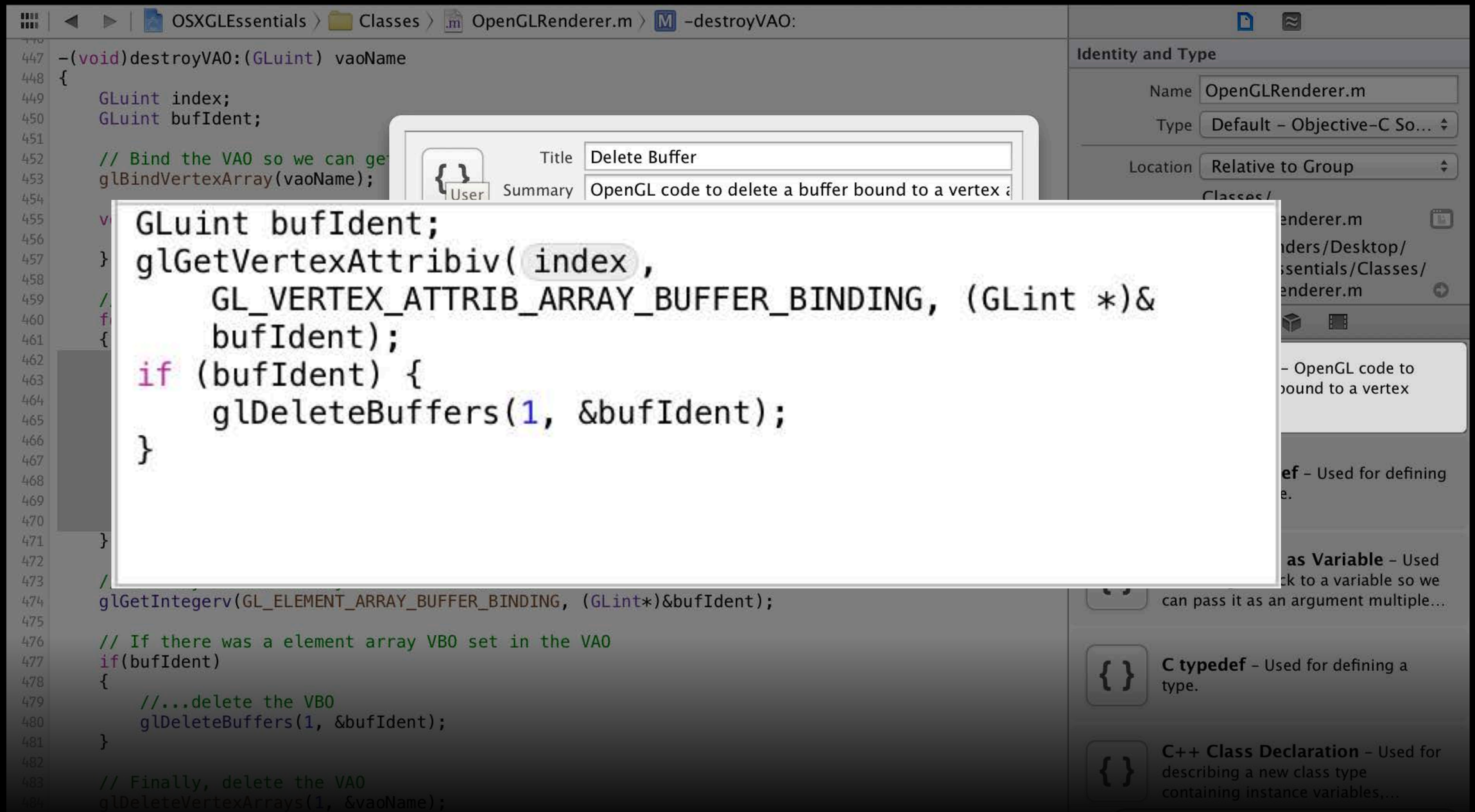
Custom Code Snippets



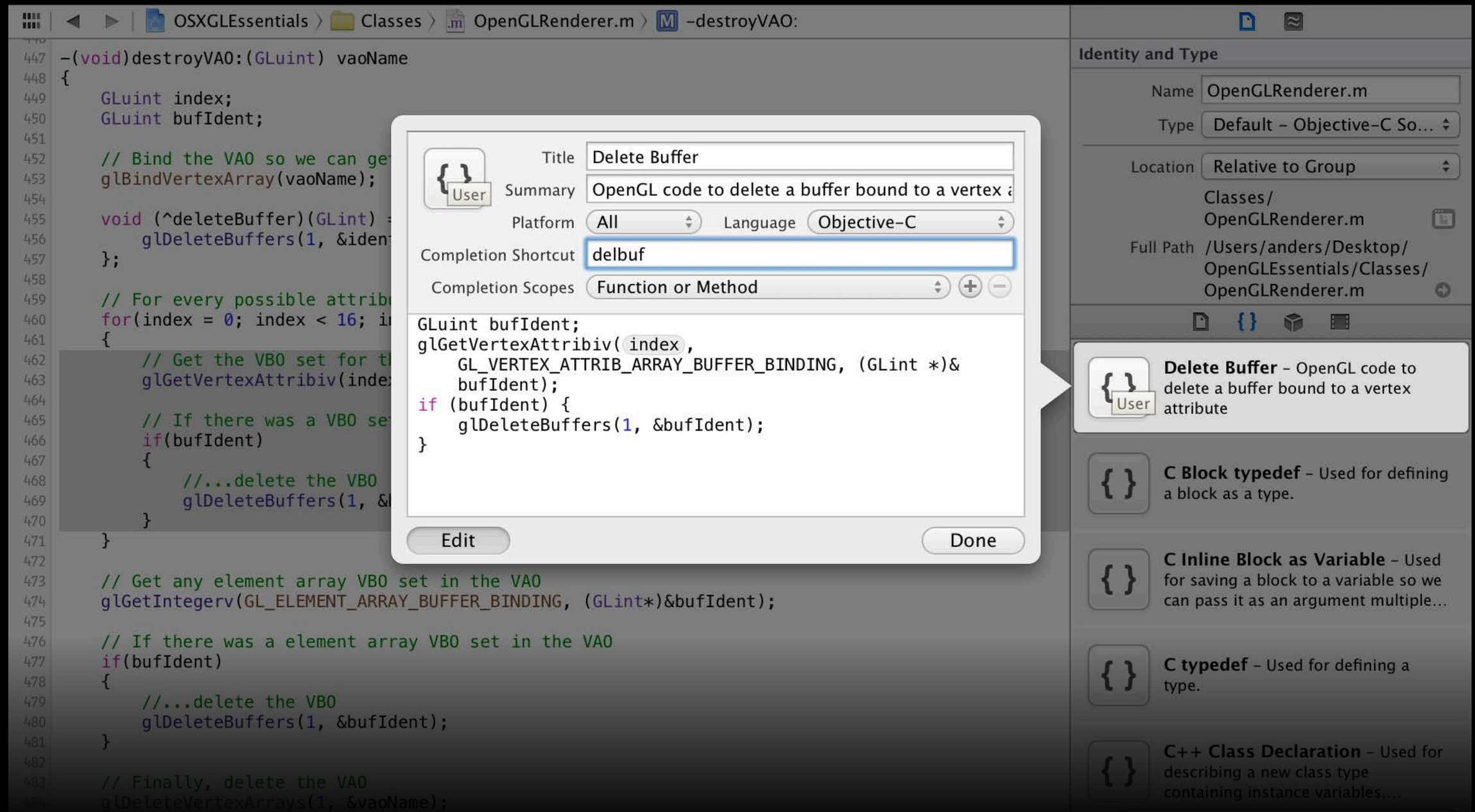
Custom Code Snippets



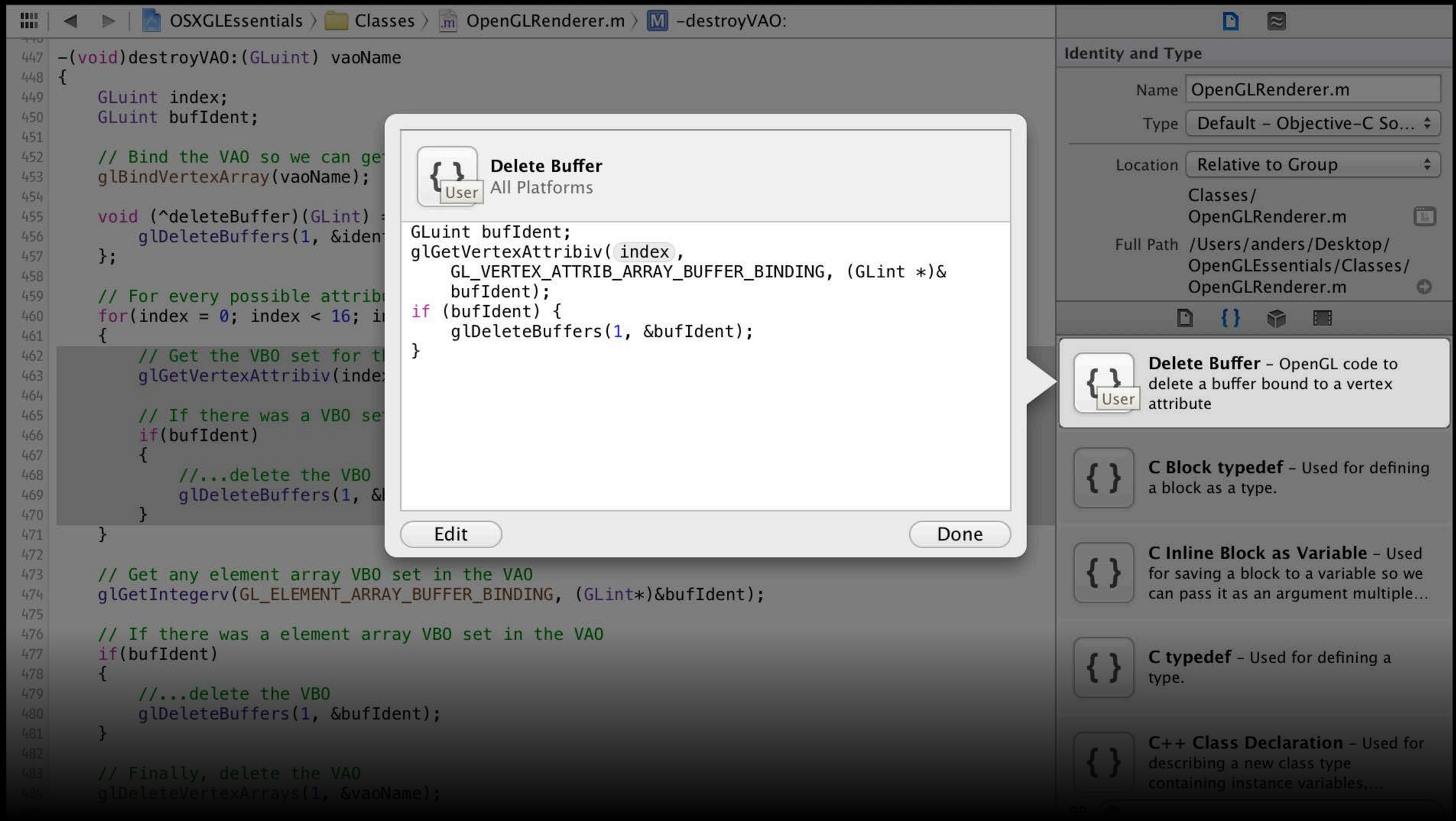
Custom Code Snippets



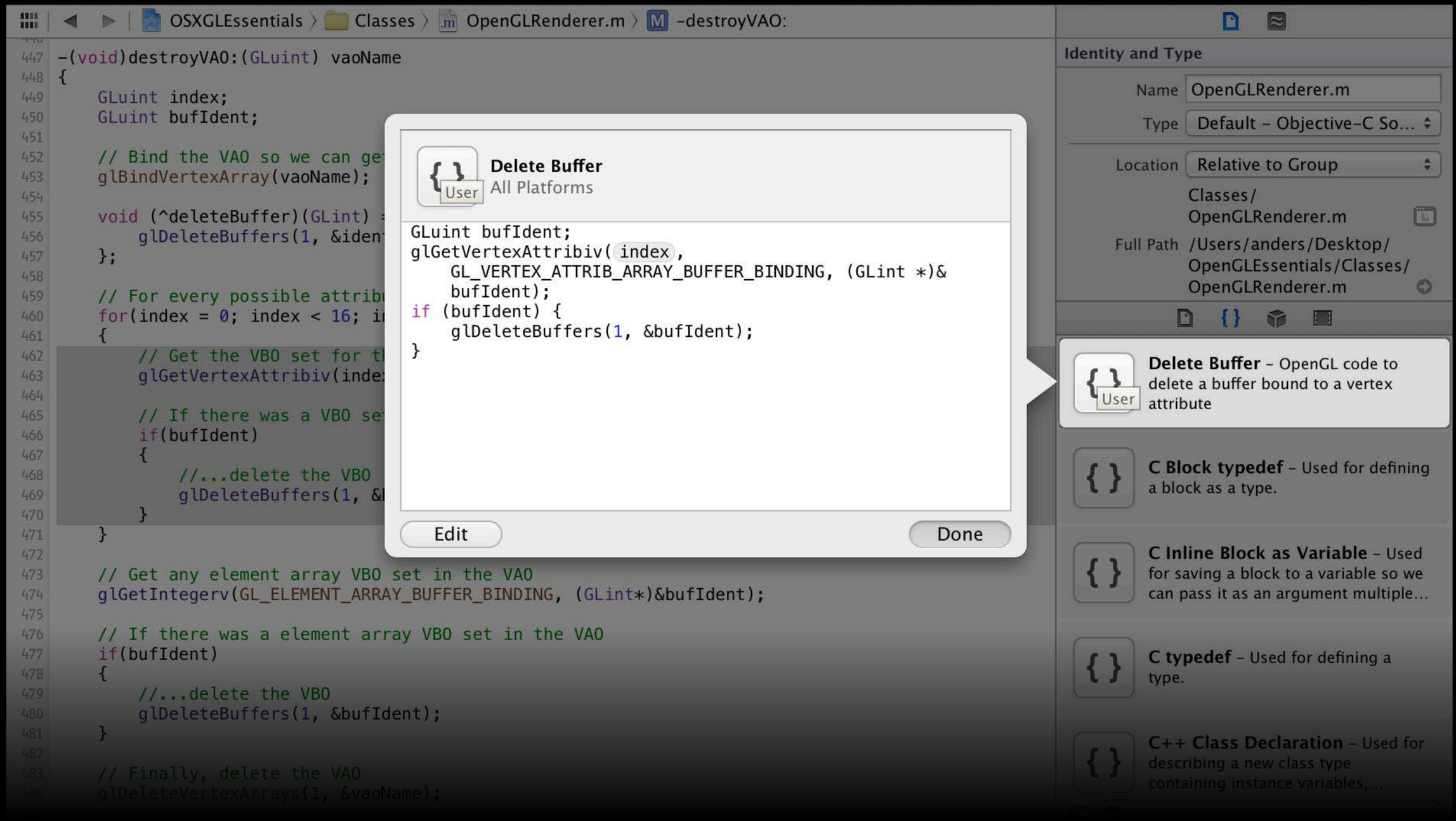
Custom Code Snippets



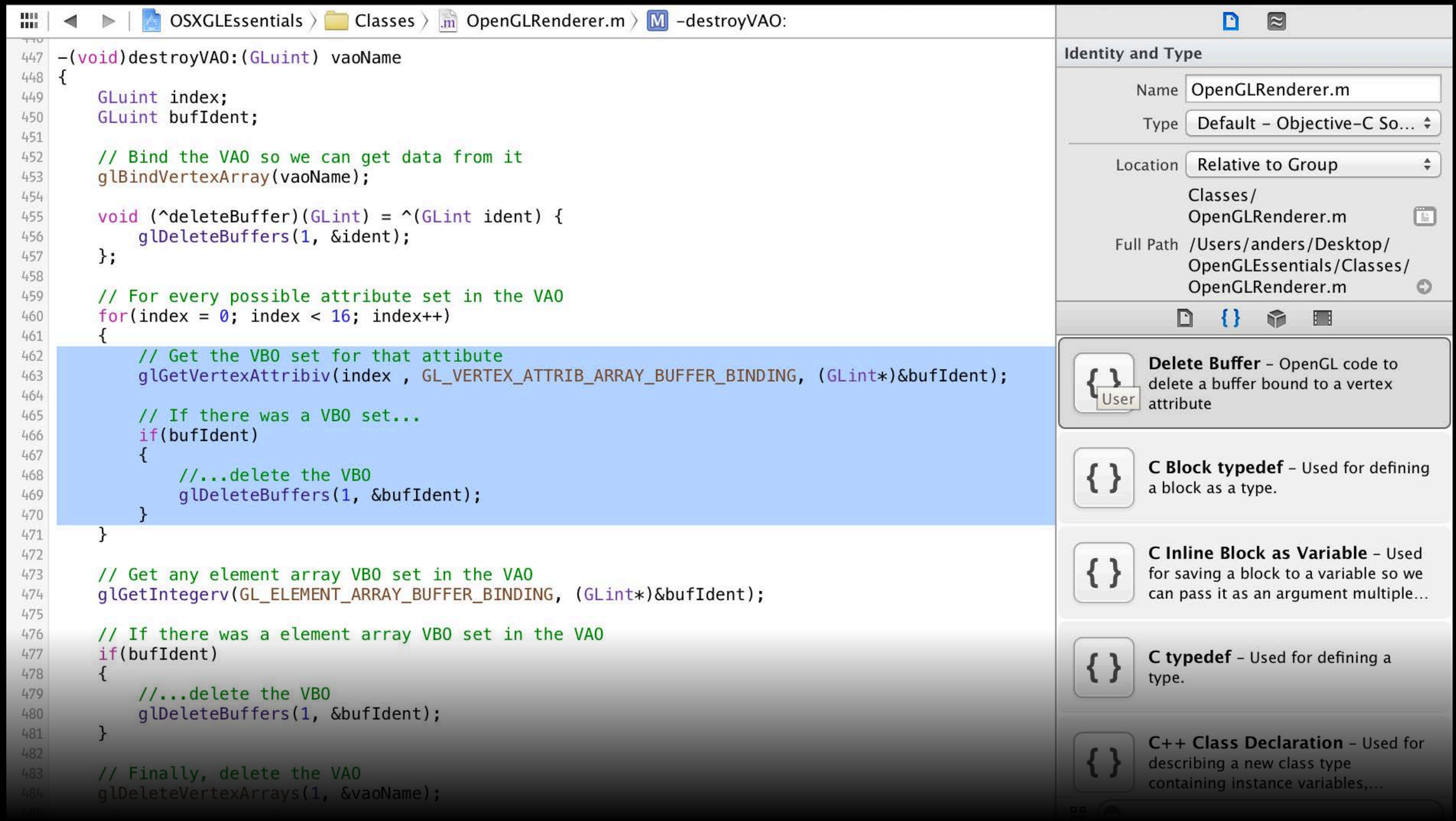
Custom Code Snippets



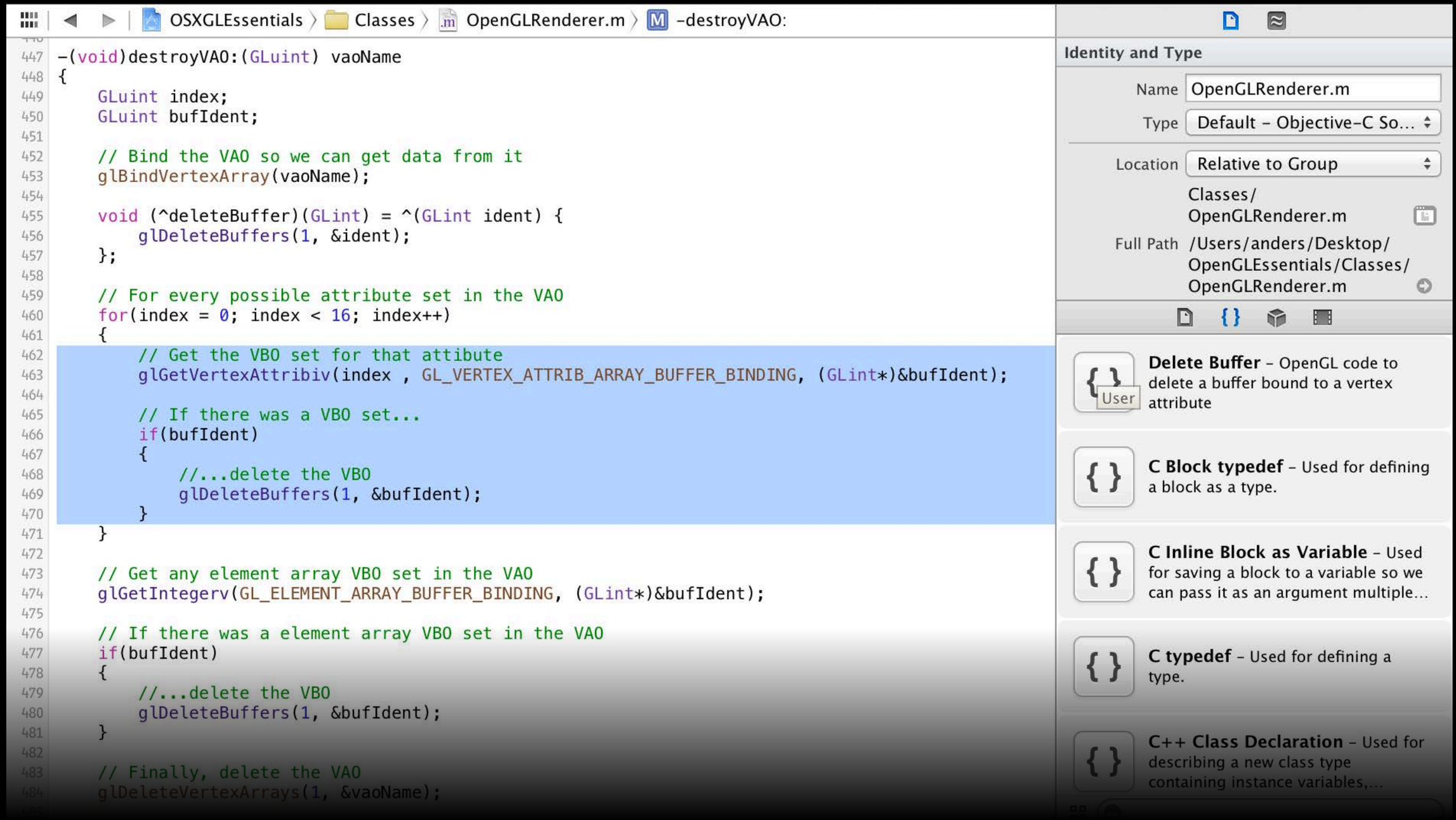
Custom Code Snippets



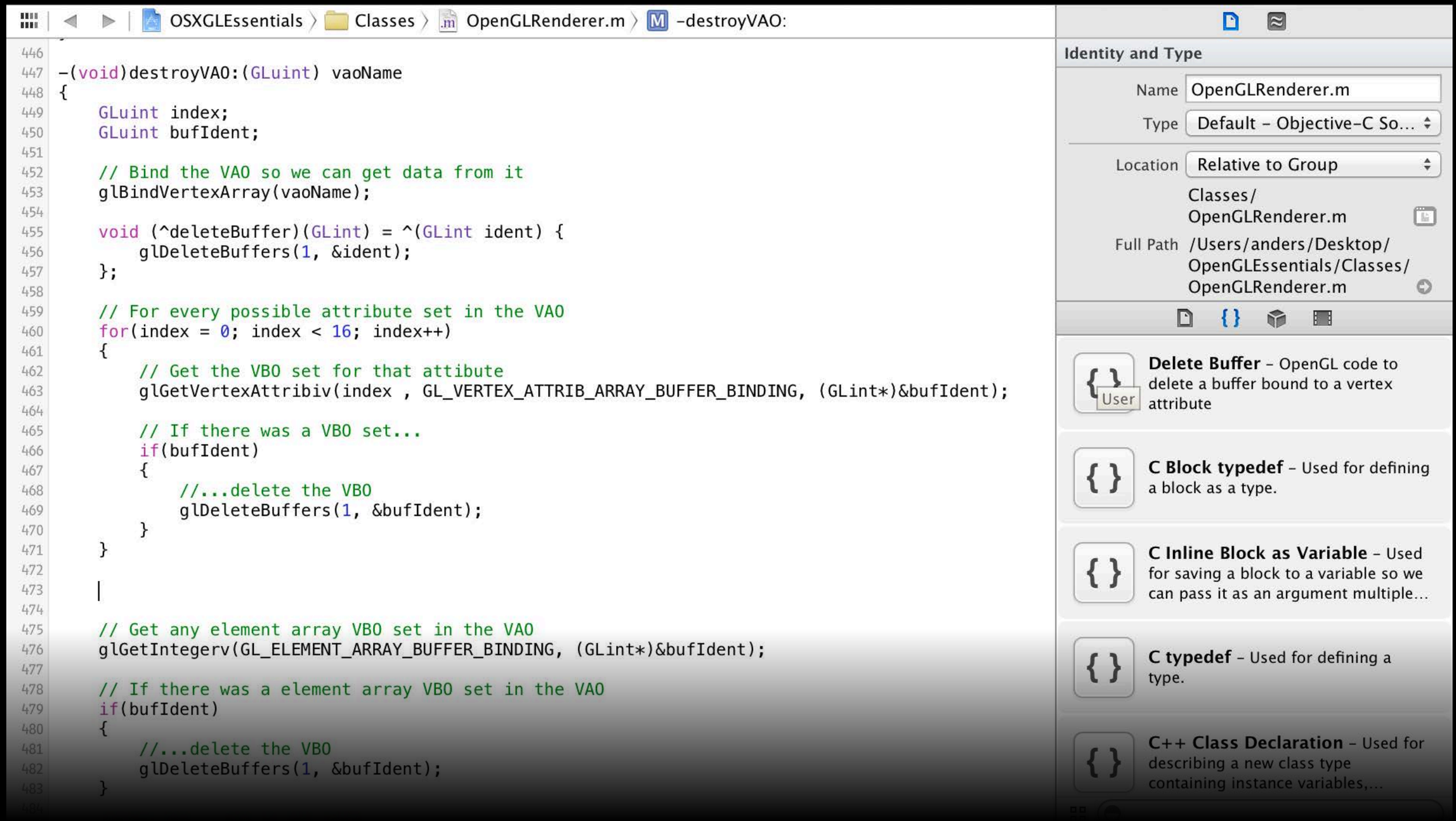
Custom Code Snippets



Custom Code Snippets



Custom Code Snippets



Custom Code Snippets

OSXGLEssentials 〉 Classes 〉 OpenGLRenderer.m 〉 -destroyVAO:

```
446
447 -(void)destroyVAO:(GLuint) vaoName
448 {
449     GLuint index;
450     GLuint bufIdent;
451
452     // Bind the VAO so we can get data from it
453     glBindVertexArray(vaoName);
454
455     void (^deleteBuffer)(GLint) = ^(GLint ident) {
456         glDeleteBuffers(1, &ident);
457     };
458
459     // For every possible attribute set in the VAO
460     for(index = 0; index < 16; index++)
461     {
462         // Get the VBO set for that attibute
463         glGetVertexAttribiv(index , GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, (GLint*)&bufIdent);
464
465         // If there was a VBO set...
466         if(bufIdent)
467         {
468             //...delete the VBO
469             glDeleteBuffers(1, &bufIdent);
470         }
471     }
472
473     delbuf
474
475
476                                                    t*)&bufIdent);
477
478     // If there was a element array VBO set in the VAO
479     if(bufIdent)
480     {
481         //...delete the VBO
482         glDeleteBuffers(1, &bufIdent);
483     }
```

delbuf – Delete Buffer

OpenGL code to delete a buffer bound to a vertex attribute

Identity and Type

Name: OpenGLRenderer.m

Type: Default – Objective-C So…

Location: Relative to Group

Classes/
OpenGLRenderer.m

Full Path: /Users/anders/Desktop/
OpenGLEssentials/Classes/
OpenGLRenderer.m

Delete Buffer – OpenGL code to delete a buffer bound to a vertex attribute

User

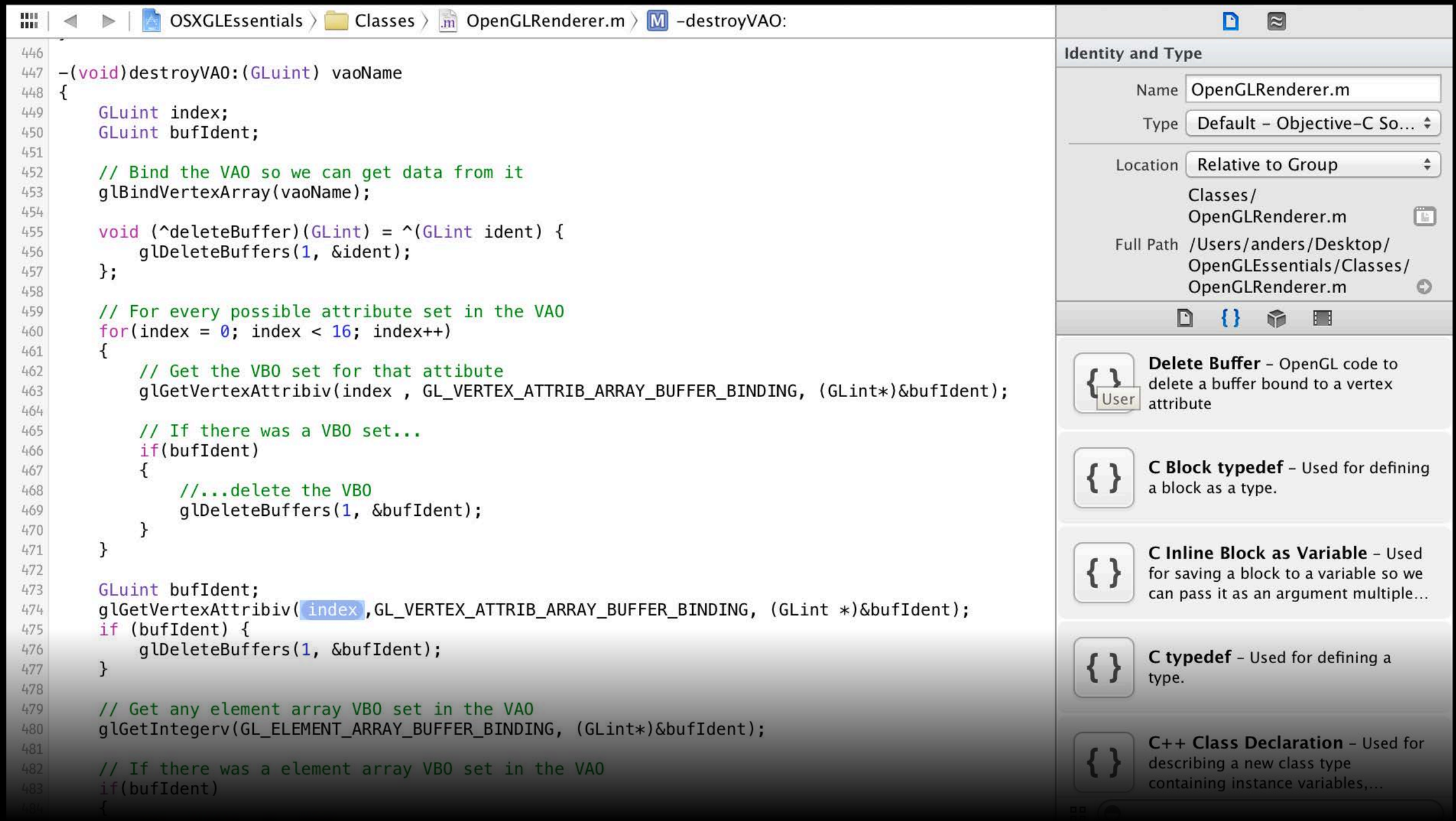
C Block typedef – Used for defining a block as a type.

C Inline Block as Variable – Used for saving a block to a variable so we can pass it as an argument multiple…

C typedef – Used for defining a type.

C++ Class Declaration – Used for describing a new class type containing instance variables,…

Custom Code Snippets



Custom Code Snippets

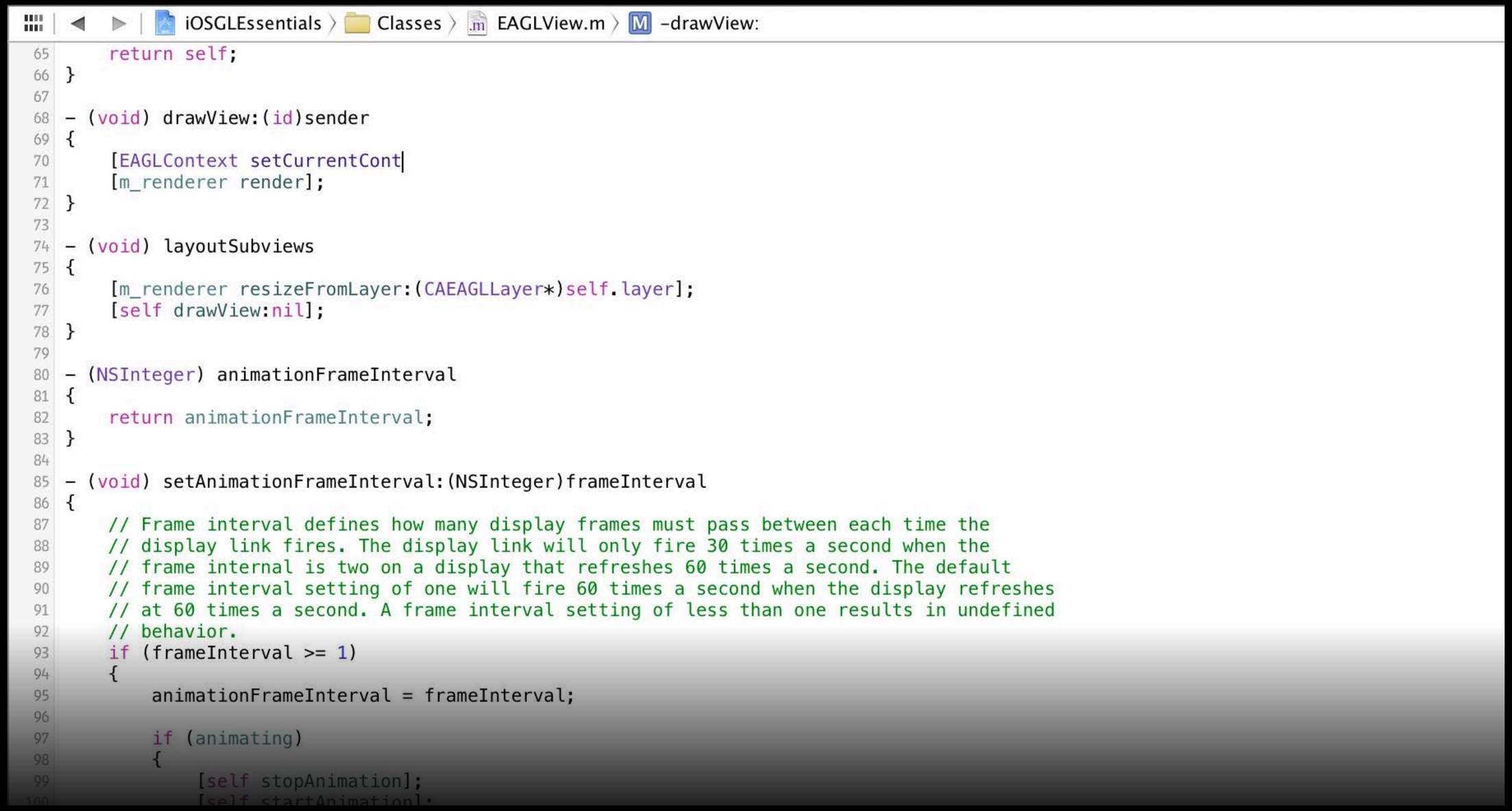
Demo

Efficient workflow

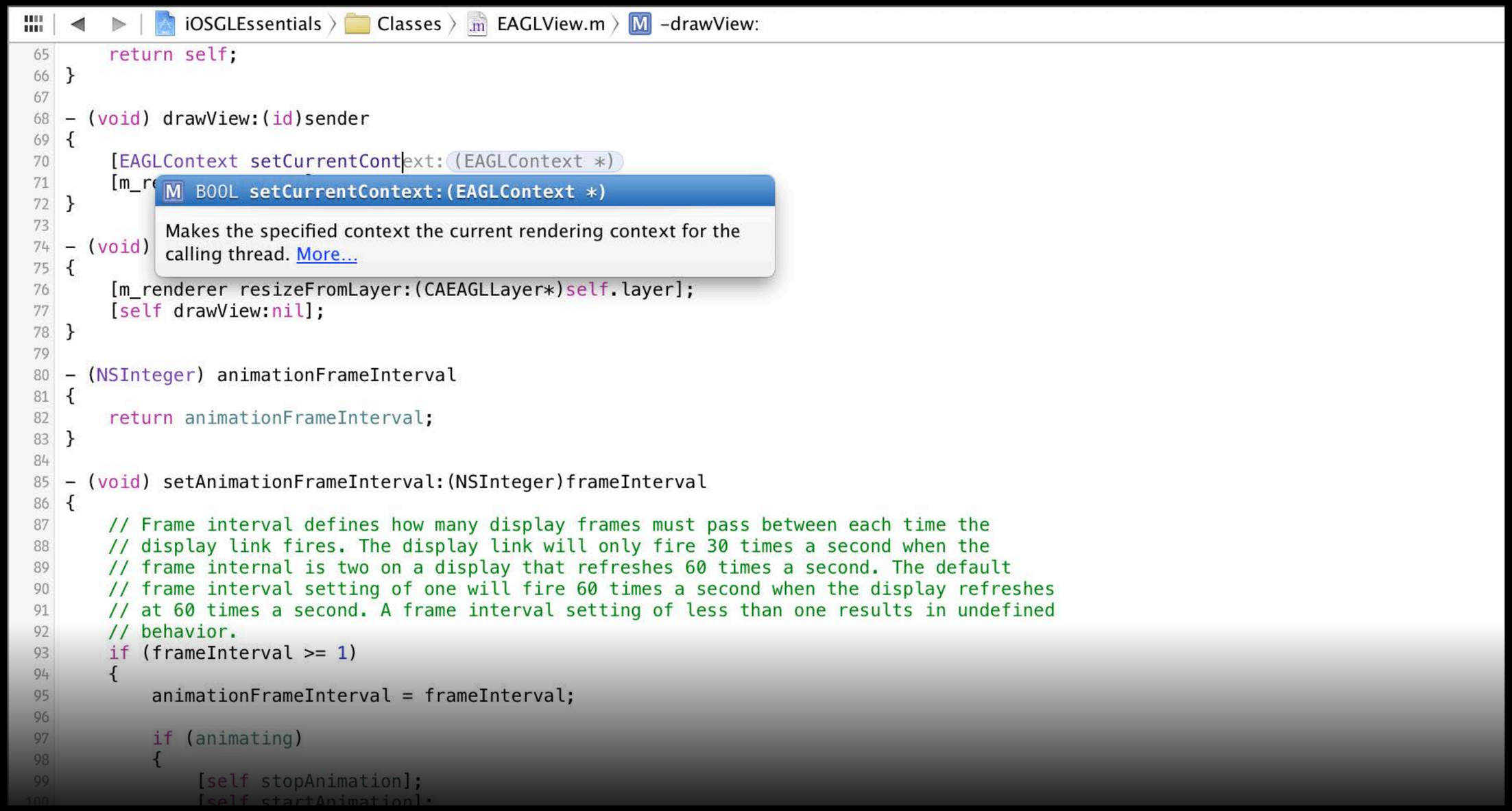
Rick Ballard

Documentation Integration

Integrated Documentation



Integrated Documentation

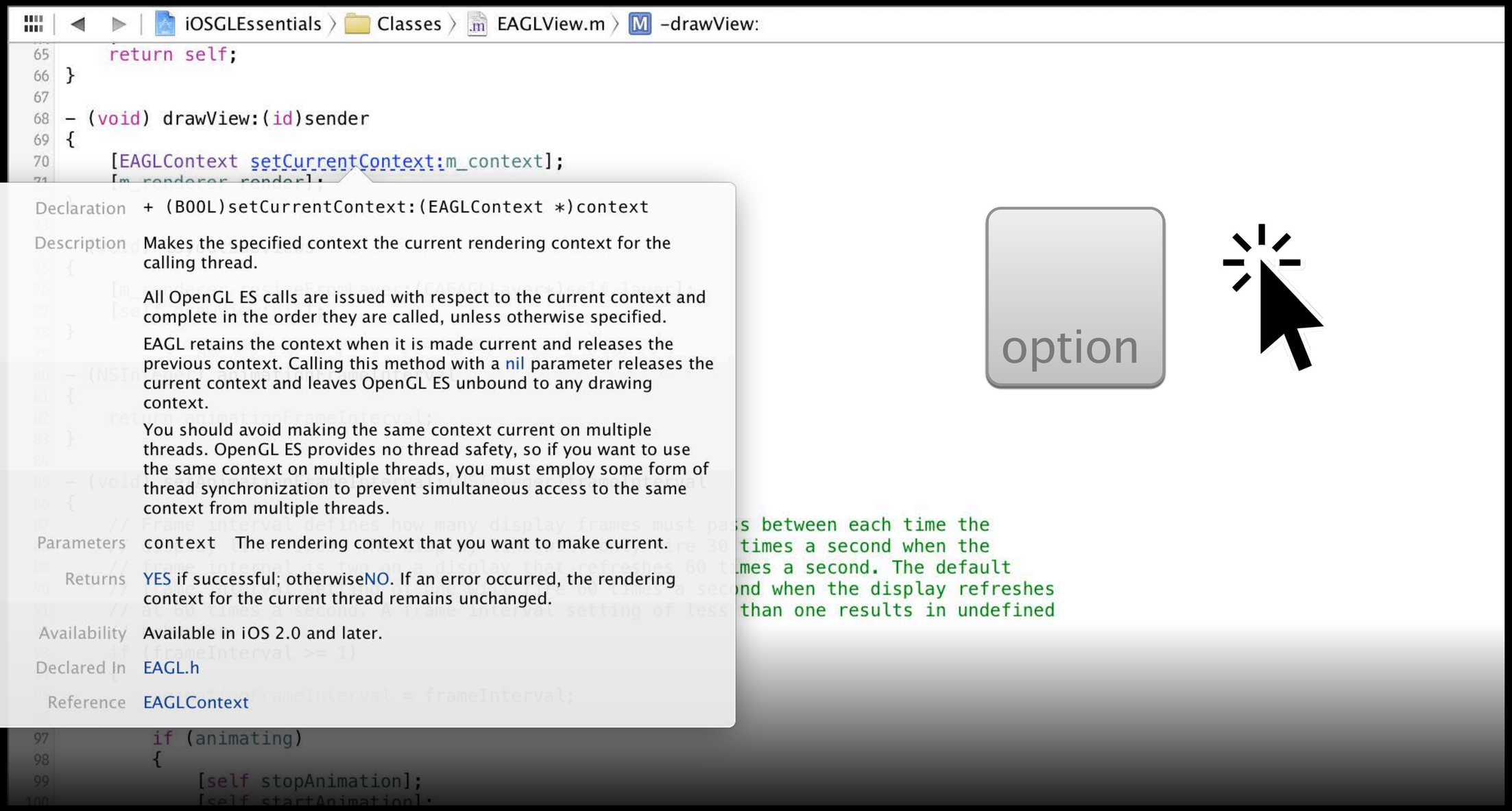


Integrated Documentation

iOSGLEssentials 〉 Classes 〉 EAGLView.m 〉 -drawView:

```
65      return self;
66  }
67
68  - (void) drawView:(id)sender
69  {
70      [EAGLContext setCurrentContext:m_context];
71      [m_renderer render];
72  }
73
74  - (void) layoutSubviews
75  {
76      [m_renderer resizeFromLayer:(CAEAGLLayer*)self.layer];
77      [self drawView:nil];
78  }
79
80  - (NSInteger) animationFrameInterval
81  {
82      return animationFrameInterval;
83  }
84
85  - (void) setAnimationFrameInterval:(NSInteger)frameInterval
86  {
87      // Frame interval defines how many display frames must pass between each time the
88      // display link fires. The display link will only fire 30 times a second when the
89      // frame internal is two on a display that refreshes 60 times a second. The default
90      // frame interval setting of one will fire 60 times a second when the display refreshes
91      // at 60 times a second. A frame interval setting of less than one results in undefined
92      // behavior.
93      if (frameInterval >= 1)
94      {
95          animationFrameInterval = frameInterval;
96
97          if (animating)
98          {
99              [self stopAnimation];
100             [self startAnimation];
```

Integrated Documentation



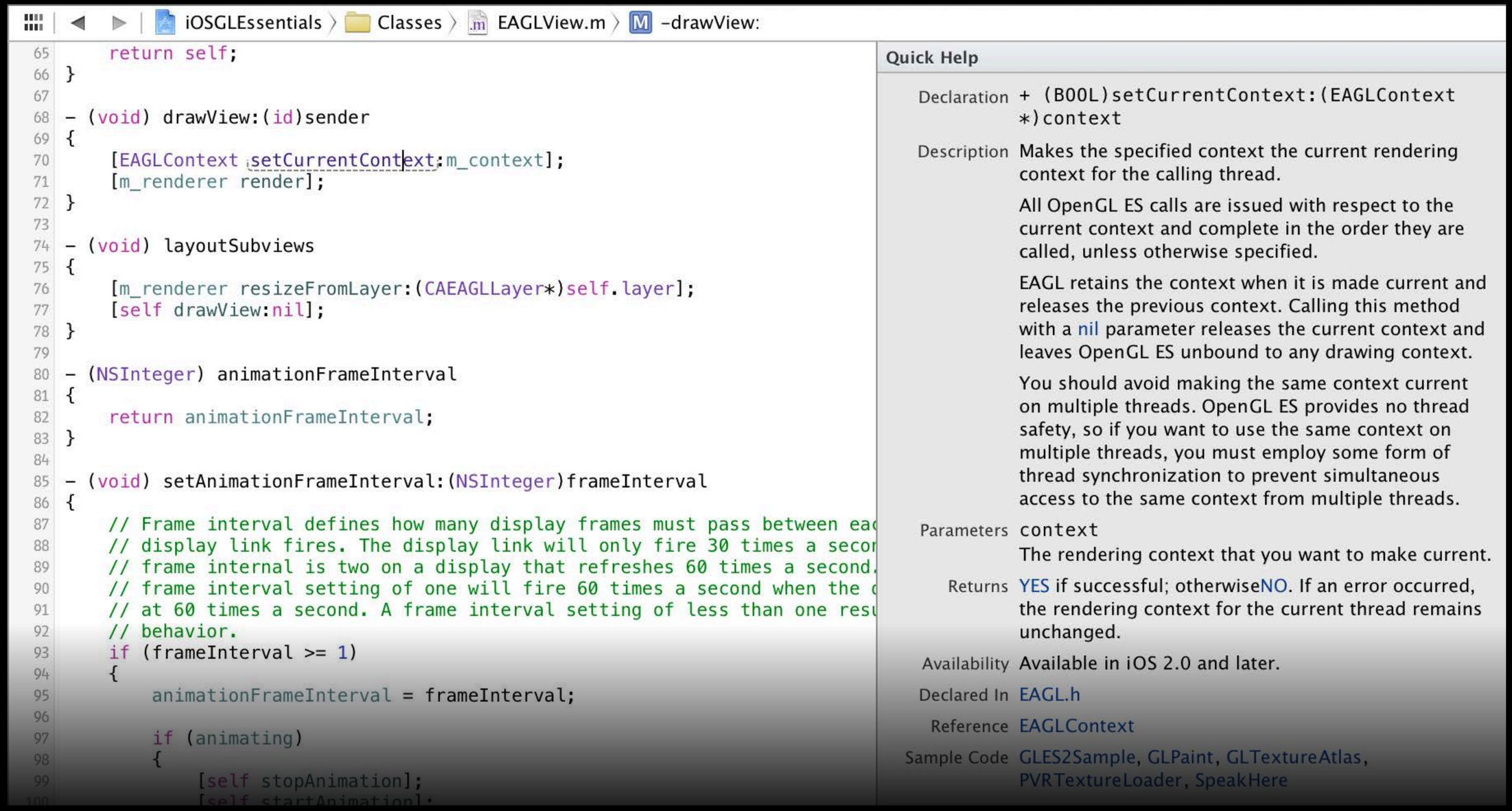
Integrated Documentation

iOSGLEssentials 〉 Classes 〉 EAGLView.m 〉 -drawView:

```
65      return self;
66  }
67
68  - (void) drawView:(id)sender
69  {
70      [EAGLContext setCurrentContext:m_context];
71      [m_renderer render];
72  }
73
74  - (void) layoutSubviews
75  {
76      [m_renderer resizeFromLayer:(CAEAGLLayer*)self.layer];
77      [self drawView:nil];
78  }
79
80  - (NSInteger) animationFrameInterval
81  {
82      return animationFrameInterval;
83  }
84
85  - (void) setAnimationFrameInterval:(NSInteger)frameInterval
86  {
87      // Frame interval defines how many display frames must pass between each time the
88      // display link fires. The display link will only fire 30 times a second when the
89      // frame internal is two on a display that refreshes 60 times a second. The default
90      // frame interval setting of one will fire 60 times a second when the display refreshes
91      // at 60 times a second. A frame interval setting of less than one results in undefined
92      // behavior.
93      if (frameInterval >= 1)
94      {
95          animationFrameInterval = frameInterval;
96
97          if (animating)
98          {
99              [self stopAnimation];
100             [self startAnimation];
```

Integrated Documentation

Integrated Documentation



Integrated Documentation

iOSGLEssentials 〉 Classes 〉 EAGLView.m 〉 -drawView:

```
65      return self;
66  }
67
68  - (void) drawView:(id)sender
69  {
70      [EAGLContext setCurrentContext:m_context];
71      [m_renderer render];
72  }
73
74  - (void) layoutSubviews
75  {
76      [m_renderer resizeFromLayer:(CAEAGLLayer*)self.layer];
77      [self drawView:nil];
78  }
79
80  - (NSInteger) animationFrameInterval
81  {
82      return animationFrameInterval;
83  }
84
85  - (void) setAnimationFrameInterval:(NSInteger)frameInterval
86  {
87      // Frame interval defines how many display frames must pass between each time the
88      // display link fires. The display link will only fire 30 times a second when the
89      // frame internal is two on a display that refreshes 60 times a second. The default
90      // frame interval setting of one will fire 60 times a second when the display refreshes
91      // at 60 times a second. A frame interval setting of less than one results in undefined
92      // behavior.
93      if (frameInterval >= 1)
94      {
95          animationFrameInterval = frameInterval;
96
97          if (animating)
98          {
99              [self stopAnimation];
100             [self startAnimation];
```

Integrated Documentation

Integrated Documentation

iOSGLEssentials 〉 Classes 〉 EAGLView.m 〉 -drawView:

```
65      return self;
66  }
67
68  - (void) drawView:(id)sender
69  {
70      [EAGLContext setCurrentContext:m_context];
71      [m_renderer render];
72  }
73
74  - (void) layoutSubviews
75  {
76      [m_renderer resizeFromLayer:(CAEAGLLayer*)self.layer];
77      [self drawView:nil];
78  }
79
80  - (NSInteger) animationFrameInterval
81  {
82      return animationFrameInterval;
83  }
84
85  - (void) setAnimationFrameInterval:(NSInteger)frameInterval
86  {
87      // Frame interval defines how many display frames must pass between each time the
88      // display link fires. The display link will only fire 30 times a second when the
89      // frame internal is two on a display that refreshes 60 times a second. The default
90      // frame interval setting of one will fire 60 times a second when the display refreshes
91      // at 60 times a second. A frame interval setting of less than one results in undefined
92      // behavior.
93      if (frameInterval >= 1)
94      {
95          animationFrameInterval = frameInterval;
96
97          if (animating)
98          {
99              [self stopAnimation];
100             [self startAnimation];
```

Works for Your Own Code Too

```
33
34 - (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;
35
```

Works for Your Own Code Too

```
33
34 - (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;
35
```

Declared In ES2Renderer.h

Works for Your Own Code Too

```
33
34 - (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;
35
```

Works for Your Own Code Too

```
26
27 /*! Resizes the receiving layer to match the dimensions of \p layer (which may not be nil),
28     and returns YES if it successfully resized the underlying framebuffer object.
29
30     @param layer  The layer to whose dimensions the receiver should resize.
31     @return       Returns YES if the framebuffer was successfully resized, NO if not.
32 */
33
34 - (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;
35
```

Works for Your Own Code Too

```
26
27  /*! Resizes the receiving layer to match the dimensions of \p layer (which may not be nil),
                YES if it successfully resized the underlying framebuffer object.

                The layer to whose dimensions the receiver should resize.
                Returns YES if the framebuffer was successfully resized, NO if not.


34  - (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;
35
```

@param
@return

Works for Your Own Code Too

```
26
27 /*! Resizes the receiving layer to match the dimensions of \p layer (which may not be nil),
28     and returns YES if it successfully resized the underlying framebuffer object.
29
30     @param layer  The layer to whose dimensions the receiver should resize.
31     @return       Returns YES if the framebuffer was successfully resized, NO if not.
32 */
33
34 - (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;
35
```

Works for Your Own Code Too

```
26
27 /*! Resizes the receiving layer to match the dimensions of \p layer (which may not be nil),
28     and returns YES if it successfully resized the underlying framebuffer object.
29
30     @param layer  The layer to whose dimensions the receiver should resize.
31     @return       Returns YES if the framebuffer was successfully resized, NO if not.
32 */
33
34 - (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;
35
```

Declaration	`- (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;`
Description	Resizes the receiving layer to match the dimensions of `layer` (which may not be nil), and returns YES if it successfully resized the underlying framebuffer object.
Parameters	`layer` The layer to whose dimensions the receiver should resize.
Returns	Returns YES if the framebuffer was successfully resized, NO if not.
Declared In	ES2Renderer.h

Works for Your Own Code Too

```
26
27 /*! Resizes the receiving layer to match the dimensions of \p layer (which may not be nil),
28     and returns YES if it successfully resized the underlying framebuffer object.
29
30     @param layer  The layer to whose dimensions the receiver should resize.
31     @return       Returns YES if the framebuffer was successfully resized, NO if not.
32 */
33
34 - (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;
35
```

Declaration `- (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;`

Description Resizes the receiving layer to match the dimensions of `layer` (which may not be nil), and returns YES if it successfully resized the underlying framebuffer object.

Parameters `layer` The layer to whose dimensions the receiver should resize.

Returns Returns YES if the framebuffer was successfully resized, NO if not.

Declared In ES2Renderer.h

Works for Your Own Code Too

```
26
27 /*! Resizes the receiving layer to match the dimensions of \p layer (which may not be nil),
28     and returns YES if it successfully resized the underlying framebuffer object.
29
30     @param layer  The layer to whose dimensions the receiver should resize.
31     @return       Returns YES if the framebuffer was successfully resized, NO if not.
32 */
33
34 - (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;
35
```

Declaration	`- (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;`
Description	Resizes the receiving layer to match the dimensions of `layer` (which may not be nil), and returns YES if it successfully resized the underlying framebuffer object.
Parameters	`layer` The layer to whose dimensions the receiver should resize.
Returns	Returns YES if the framebuffer was successfully resized, NO if not.
Declared In	ES2Renderer.h

Works for Your Own Code Too

```
26
27 /*! Resizes the receiving layer to match the dimensions of \p layer (which may not be nil),
28     and returns YES if it successfully resized the underlying framebuffer object.
29
30     @param layer  The layer to whose dimensions the receiver should resize.
31     @return       Returns YES if the framebuffer was successfully resized, NO if not.
32 */
33
34 - (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;
35
```

Declaration	`- (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;`
Description	Resizes the receiving layer to match the dimensions of `layer` (which may not be nil), and returns YES if it successfully resized the underlying framebuffer object.
Parameters	`layer` The layer to whose dimensions the receiver should resize.
Returns	Returns YES if the framebuffer was successfully resized, NO if not.
Declared In	ES2Renderer.h

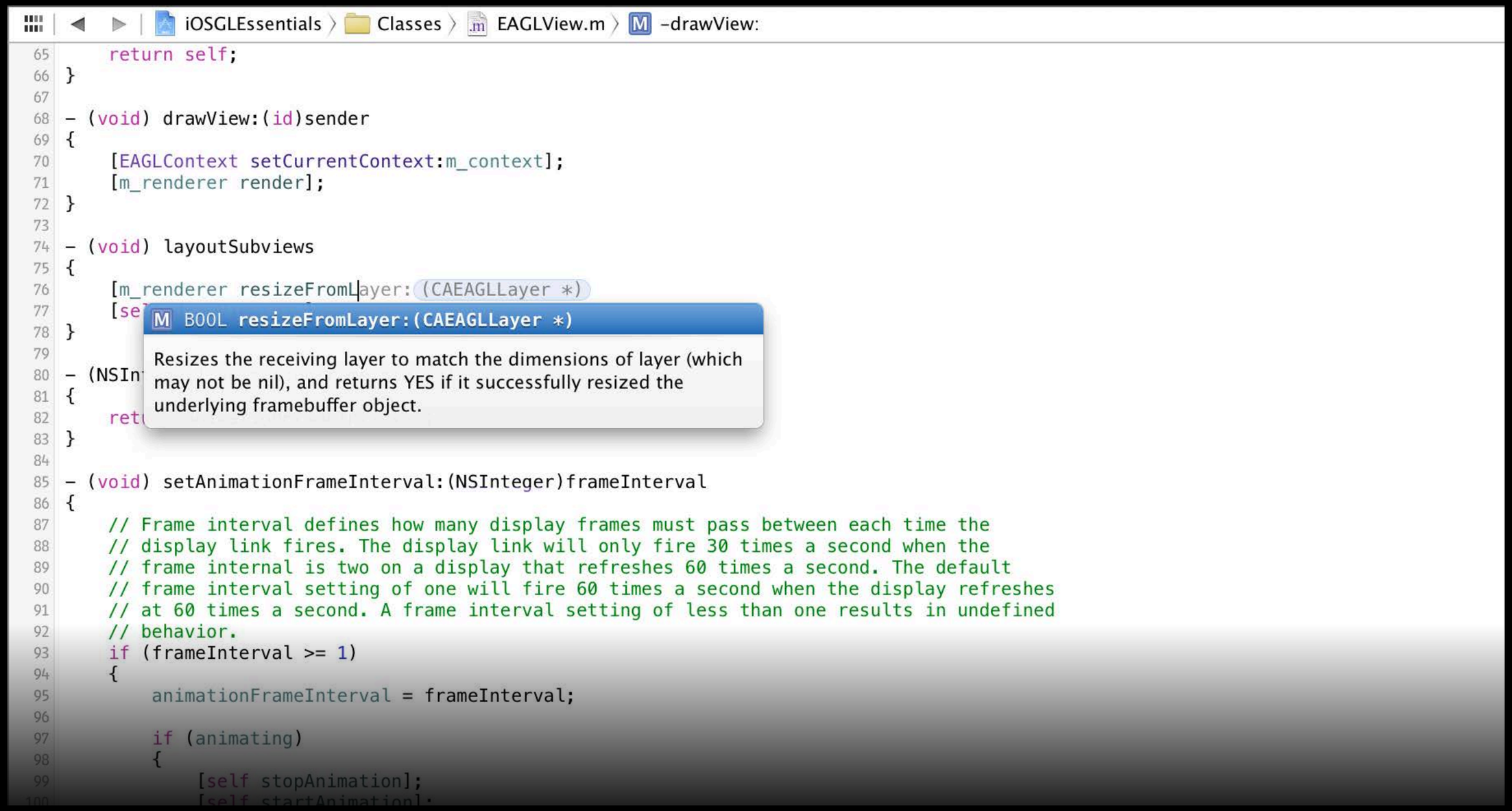
Works for Your Own Code Too

```
26
27 /*! Resizes the receiving layer to match the dimensions of \p layer (which may not be nil),
28     and returns YES if it successfully resized the underlying framebuffer object.
29
30     @param layer  The layer to whose dimensions the receiver should resize.
31     @return       Returns YES if the framebuffer was successfully resized, NO if not.
32 */
33
34 - (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;
35
```

www.doxygen.org

Declaration	`- (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;`
Description	Resizes the receiving layer to match the dimensions of `layer` (which may not be nil), and returns YES if it successfully resized the underlying framebuffer object.
Parameters	`layer` The layer to whose dimensions the receiver should resize.
Returns	Returns YES if the framebuffer was successfully resized, NO if not.
Declared In	ES2Renderer.h

Integrated Documentation

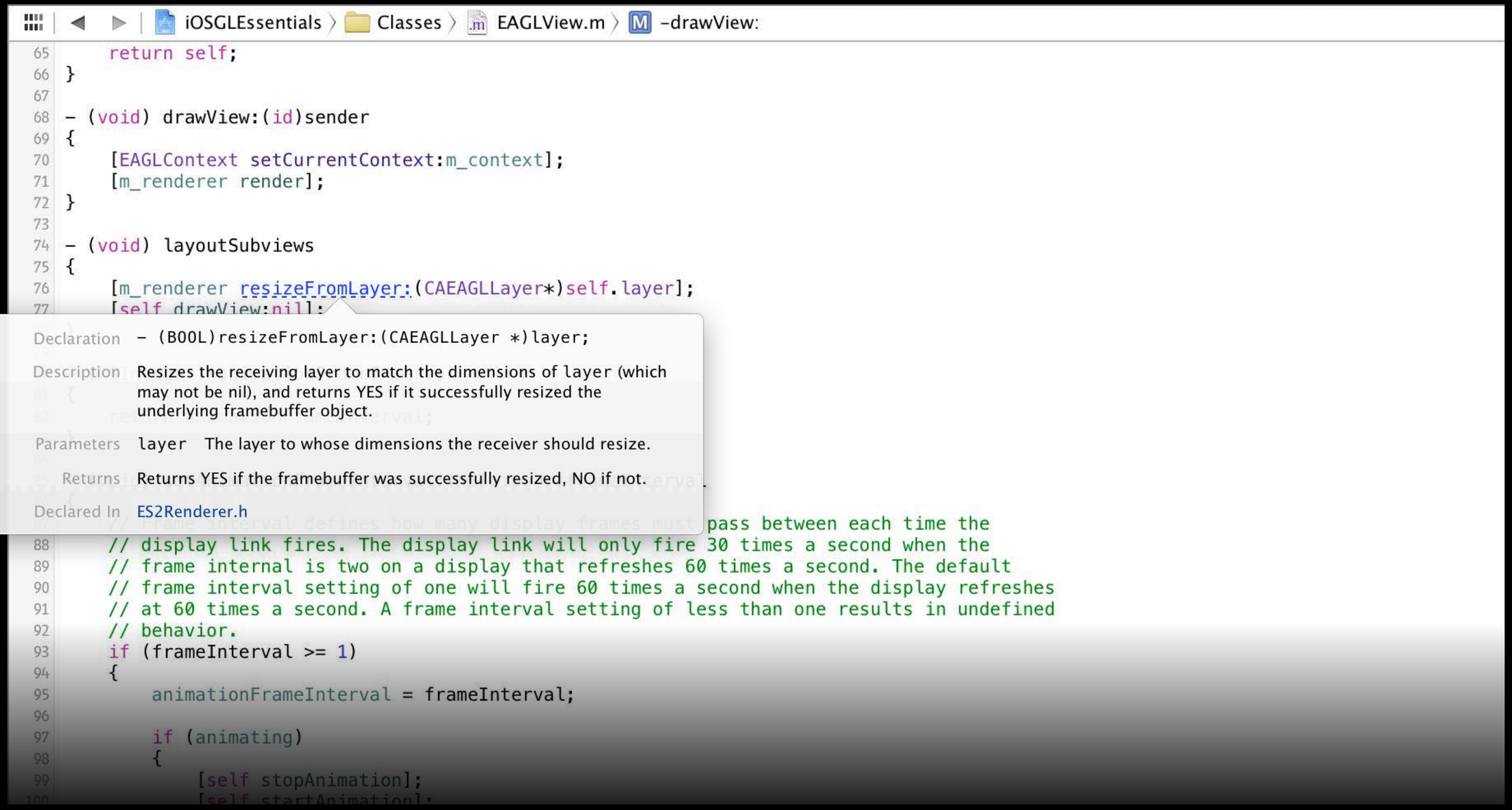


Integrated Documentation

iOSGLEssentials 〉 Classes 〉 EAGLView.m 〉 -drawView:

```
65      return self;
66  }
67
68  - (void) drawView:(id)sender
69  {
70      [EAGLContext setCurrentContext:m_context];
71      [m_renderer render];
72  }
73
74  - (void) layoutSubviews
75  {
76      [m_renderer resizeFromLayer:(CAEAGLLayer*)self.layer];
77      [self drawView:nil];
78  }
79
80  - (NSInteger) animationFrameInterval
81  {
82      return animationFrameInterval;
83  }
84
85  - (void) setAnimationFrameInterval:(NSInteger)frameInterval
86  {
87      // Frame interval defines how many display frames must pass between each time the
88      // display link fires. The display link will only fire 30 times a second when the
89      // frame internal is two on a display that refreshes 60 times a second. The default
90      // frame interval setting of one will fire 60 times a second when the display refreshes
91      // at 60 times a second. A frame interval setting of less than one results in undefined
92      // behavior.
93      if (frameInterval >= 1)
94      {
95          animationFrameInterval = frameInterval;
96
97          if (animating)
98          {
99              [self stopAnimation];
100             [self startAnimation];
```

Integrated Documentation



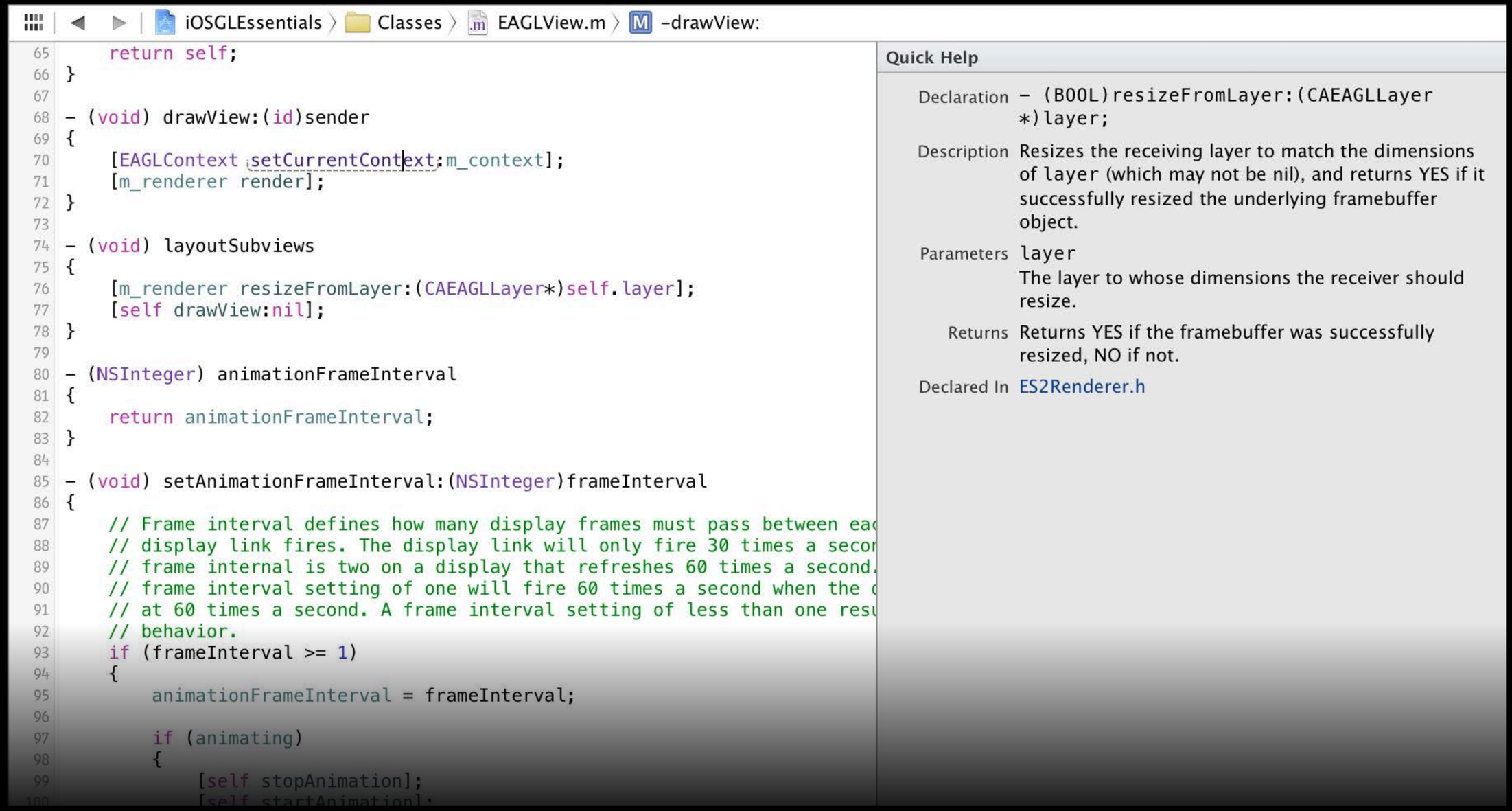
Integrated Documentation

iOSGLEssentials 〉 Classes 〉 EAGLView.m 〉 -drawView:

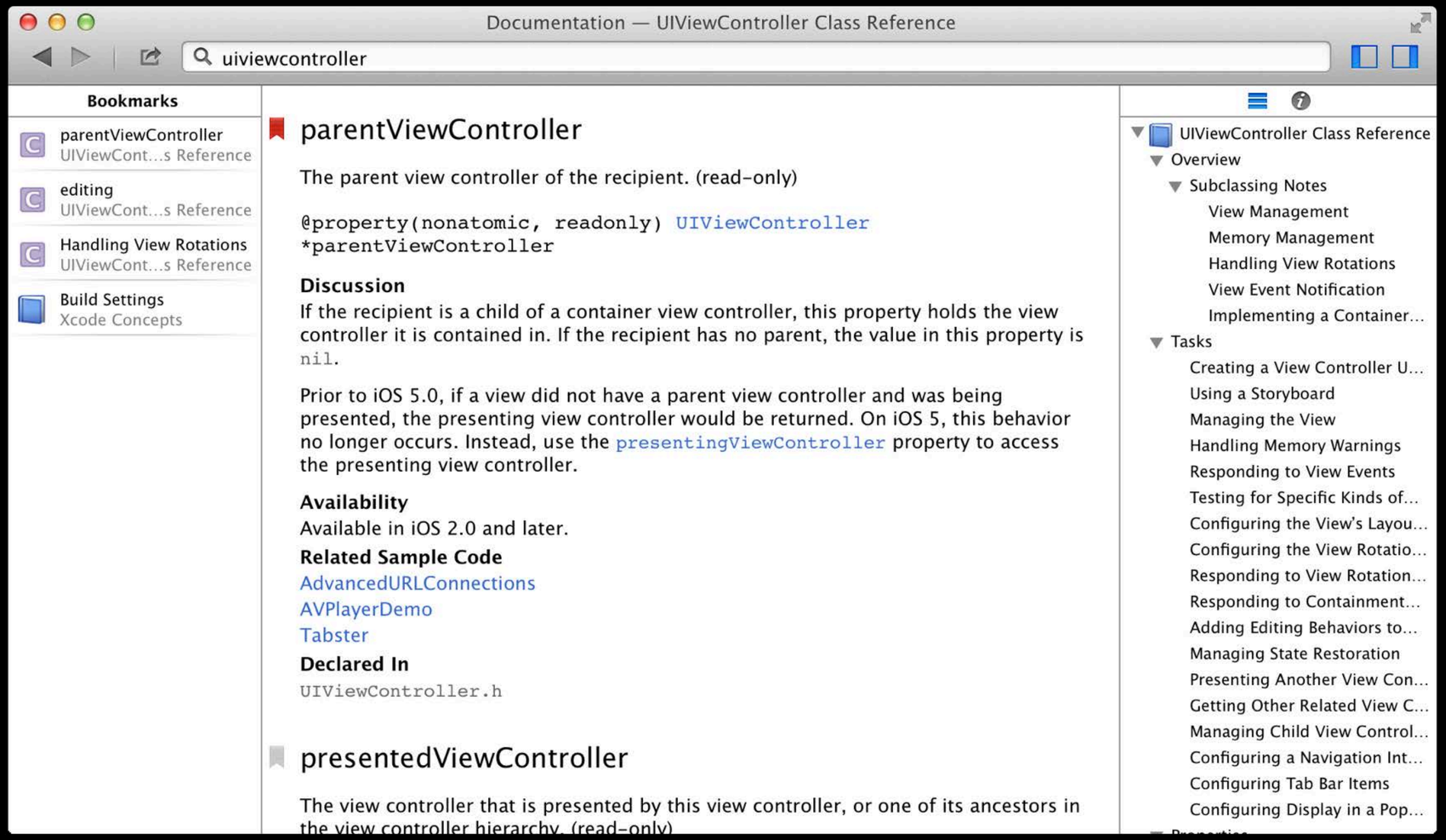
```
65      return self;
66  }
67
68  - (void) drawView:(id)sender
69  {
70      [EAGLContext setCurrentContext:m_context];
71      [m_renderer render];
72  }
73
74  - (void) layoutSubviews
75  {
76      [m_renderer resizeFromLayer:(CAEAGLLayer*)self.layer];
77      [self drawView:nil];
78  }
79
80  - (NSInteger) animationFrameInterval
81  {
82      return animationFrameInterval;
83  }
84
85  - (void) setAnimationFrameInterval:(NSInteger)frameInterval
86  {
87      // Frame interval defines how many display frames must pass between each time the
88      // display link fires. The display link will only fire 30 times a second when the
89      // frame internal is two on a display that refreshes 60 times a second. The default
90      // frame interval setting of one will fire 60 times a second when the display refreshes
91      // at 60 times a second. A frame interval setting of less than one results in undefined
92      // behavior.
93      if (frameInterval >= 1)
94      {
95          animationFrameInterval = frameInterval;
96
97          if (animating)
98          {
99              [self stopAnimation];
100             [self startAnimation];
```

Integrated Documentation

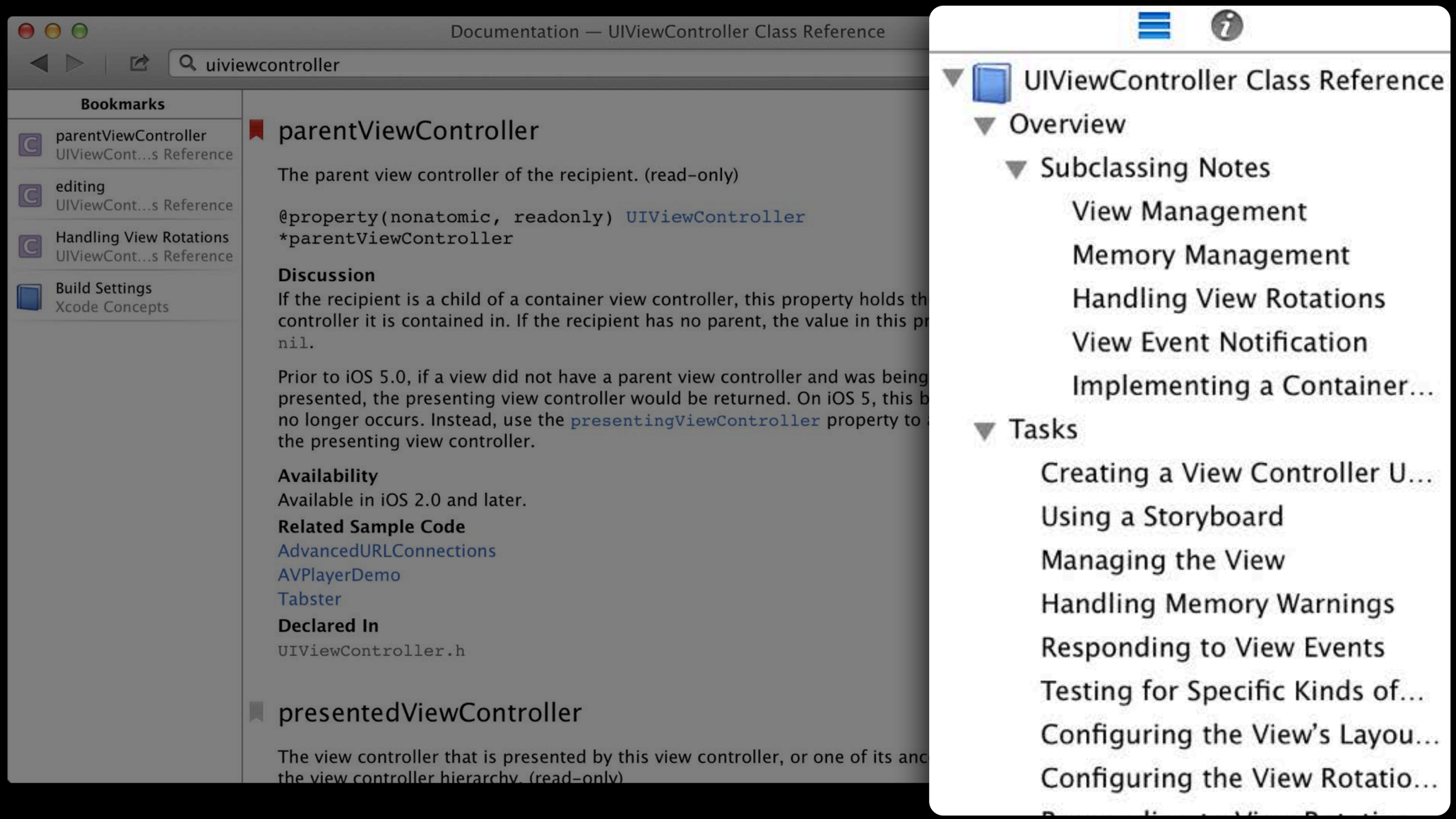
Integrated Documentation



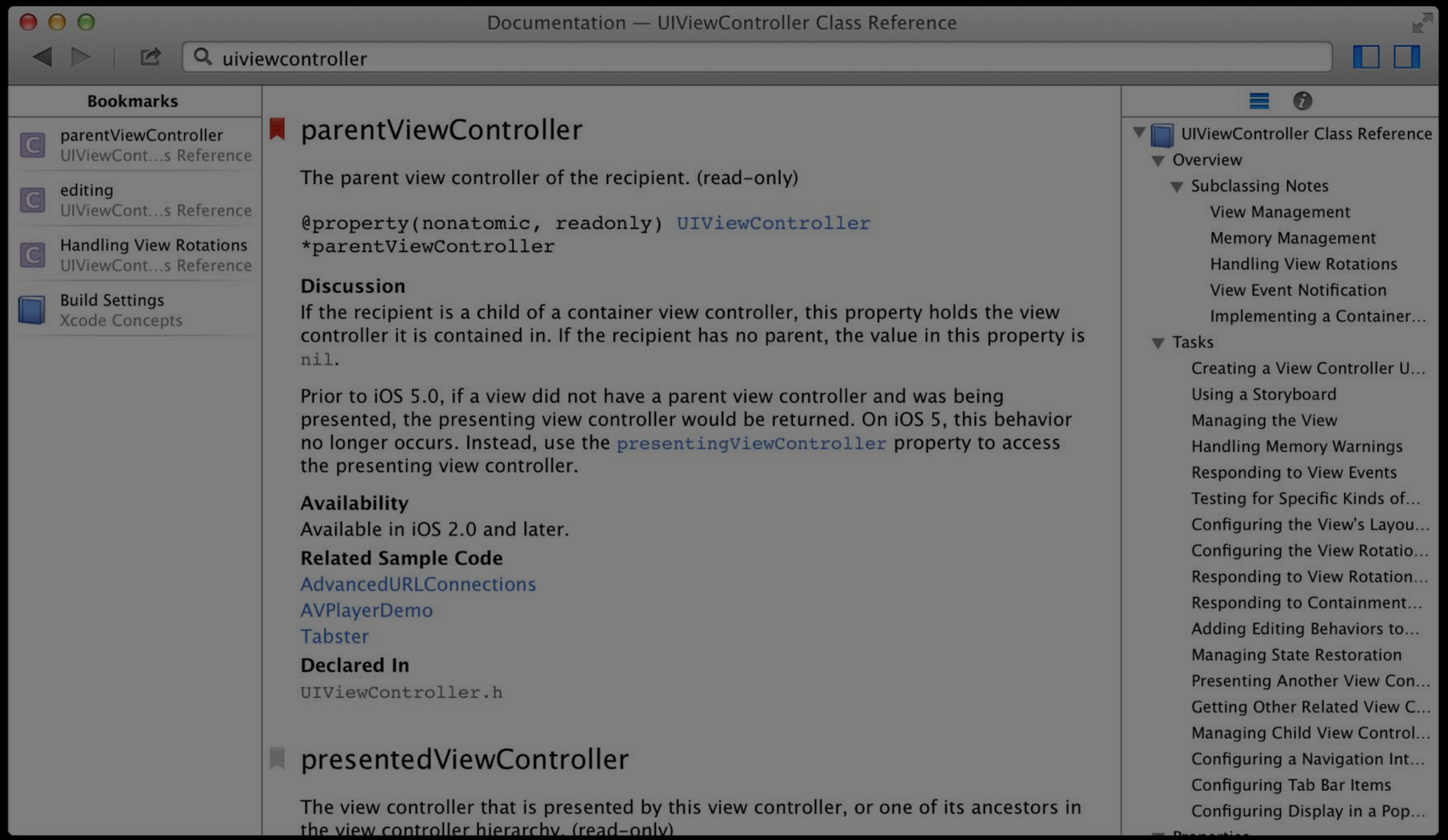
Documentation Window



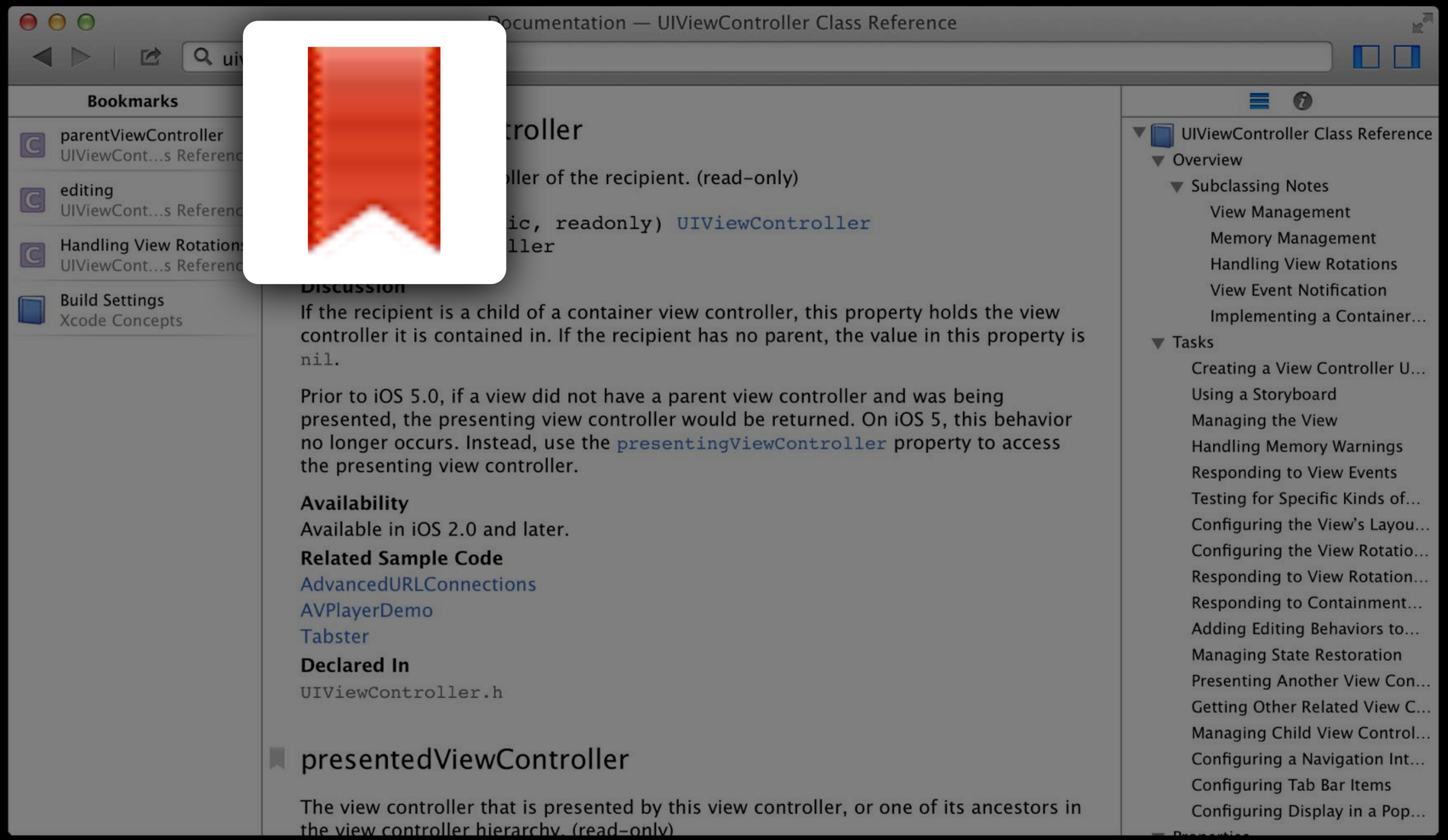
Documentation Window



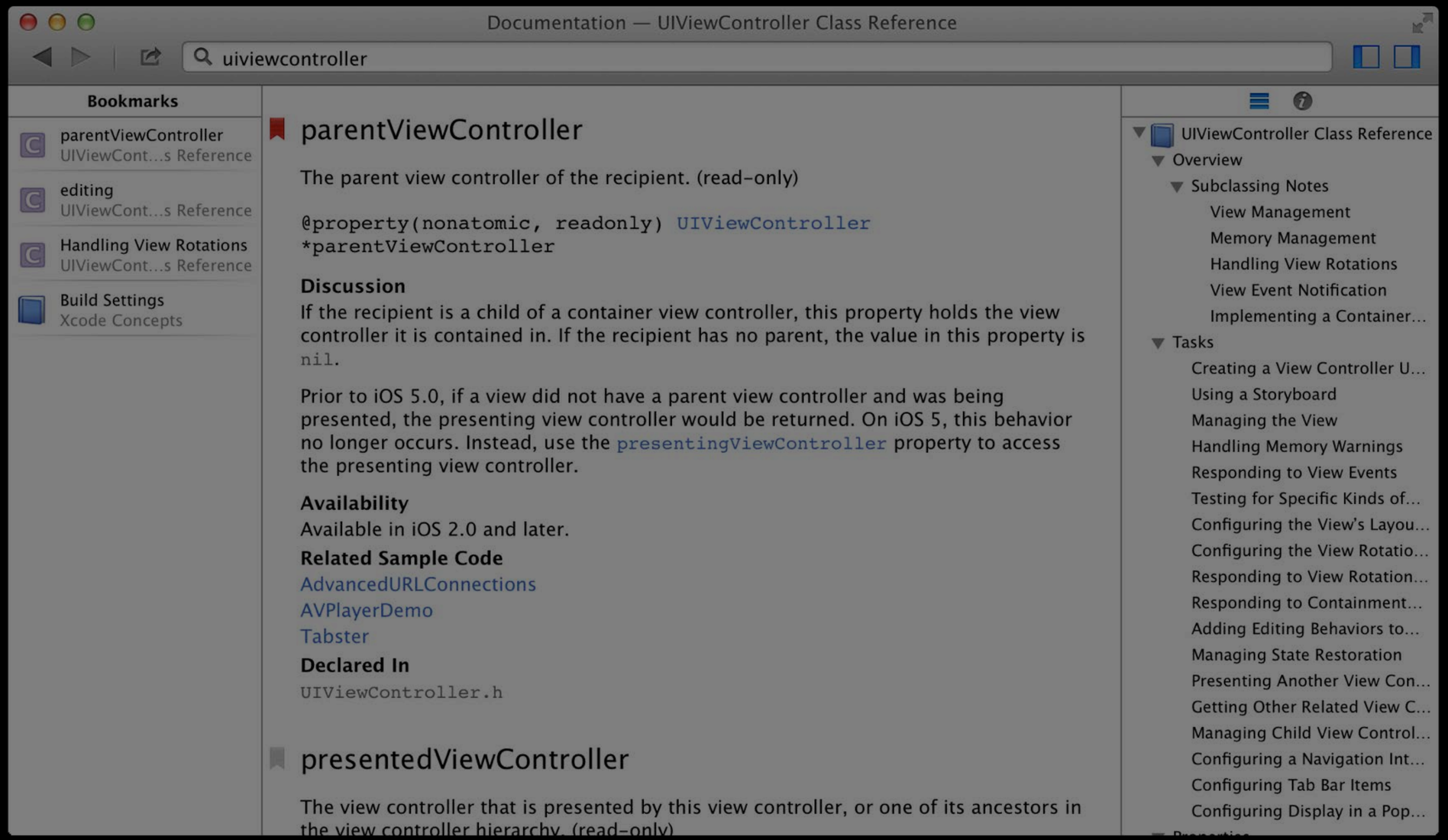
Documentation Window



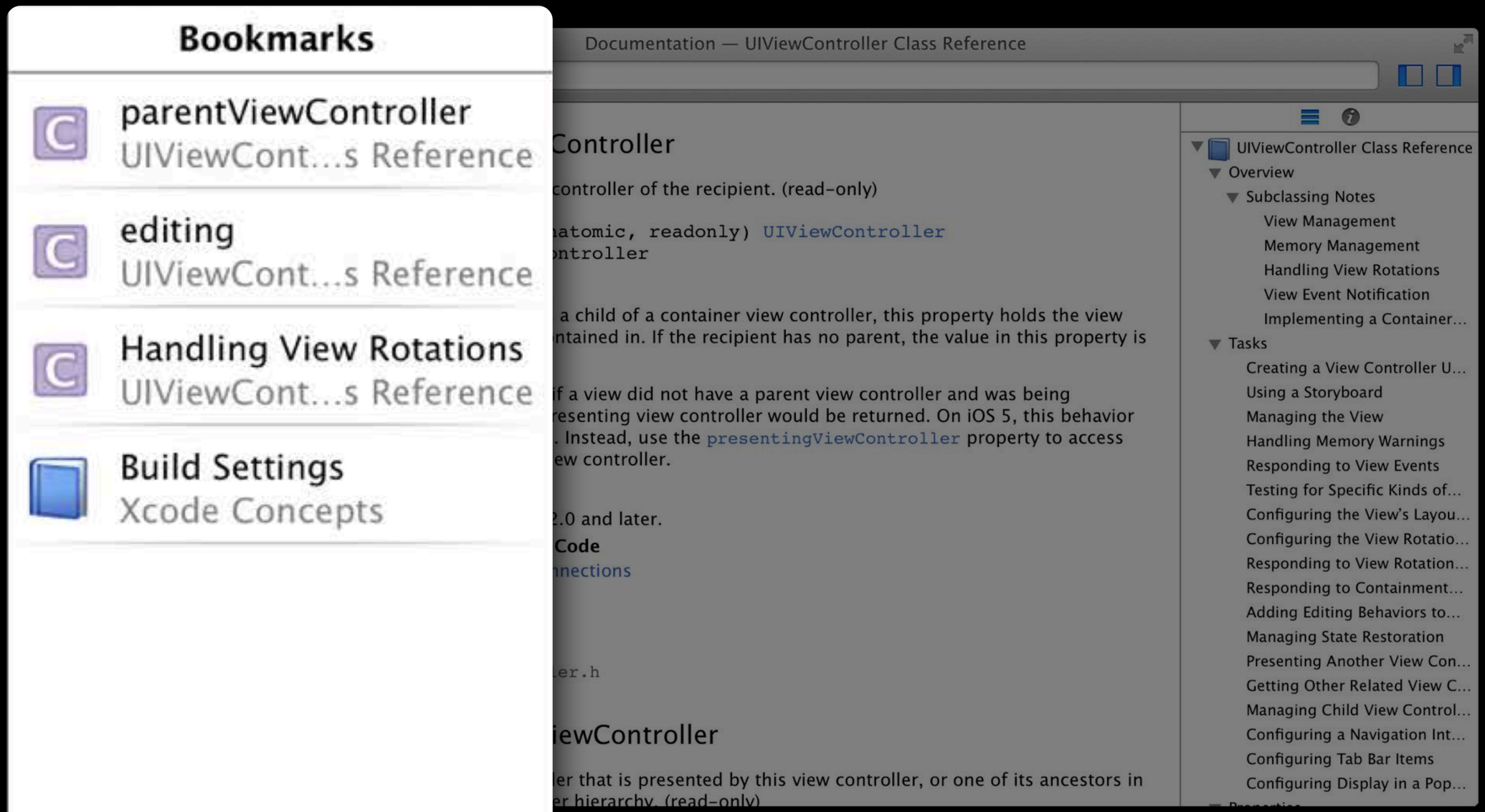
Documentation Window



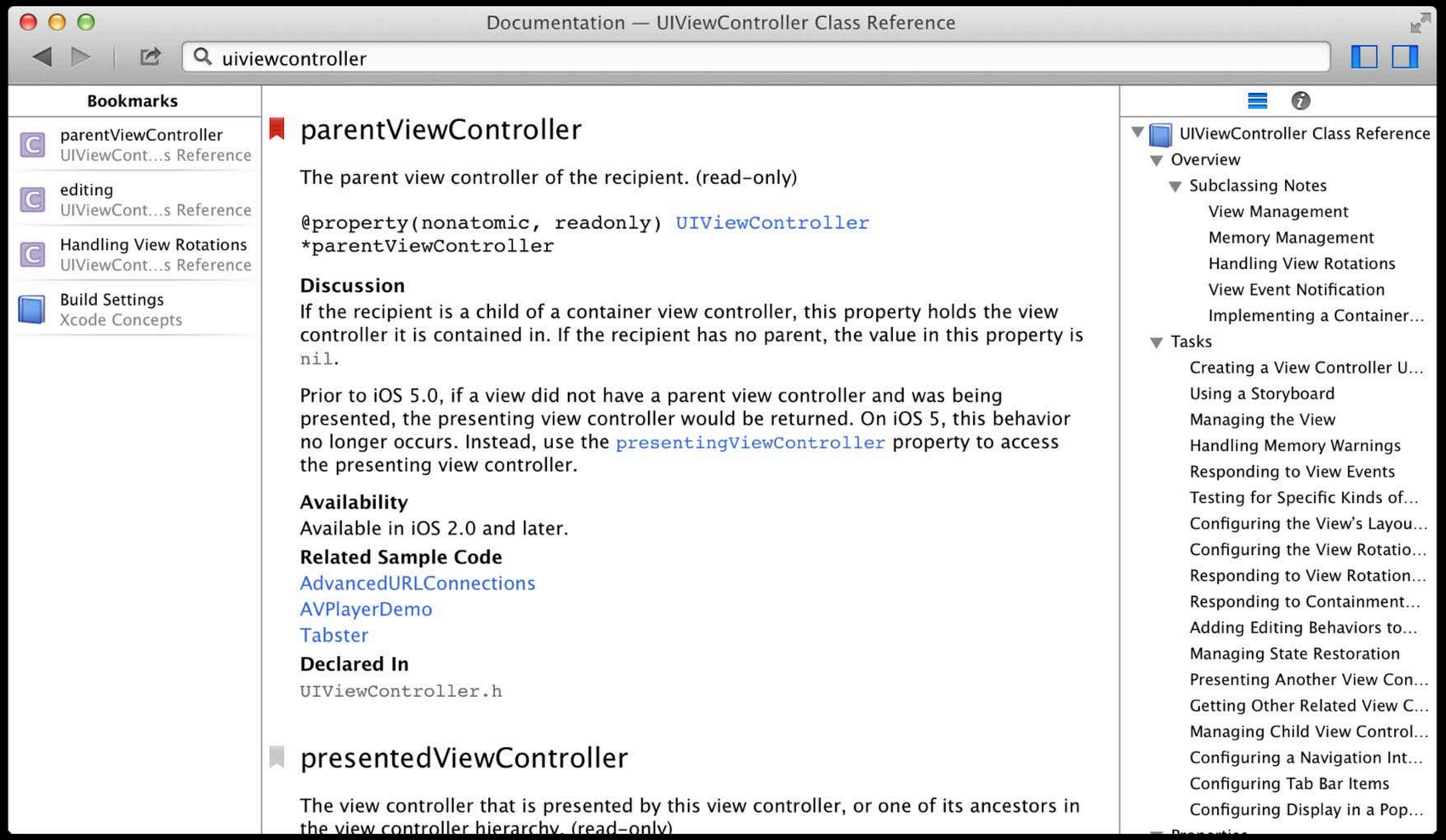
Documentation Window



Documentation Window

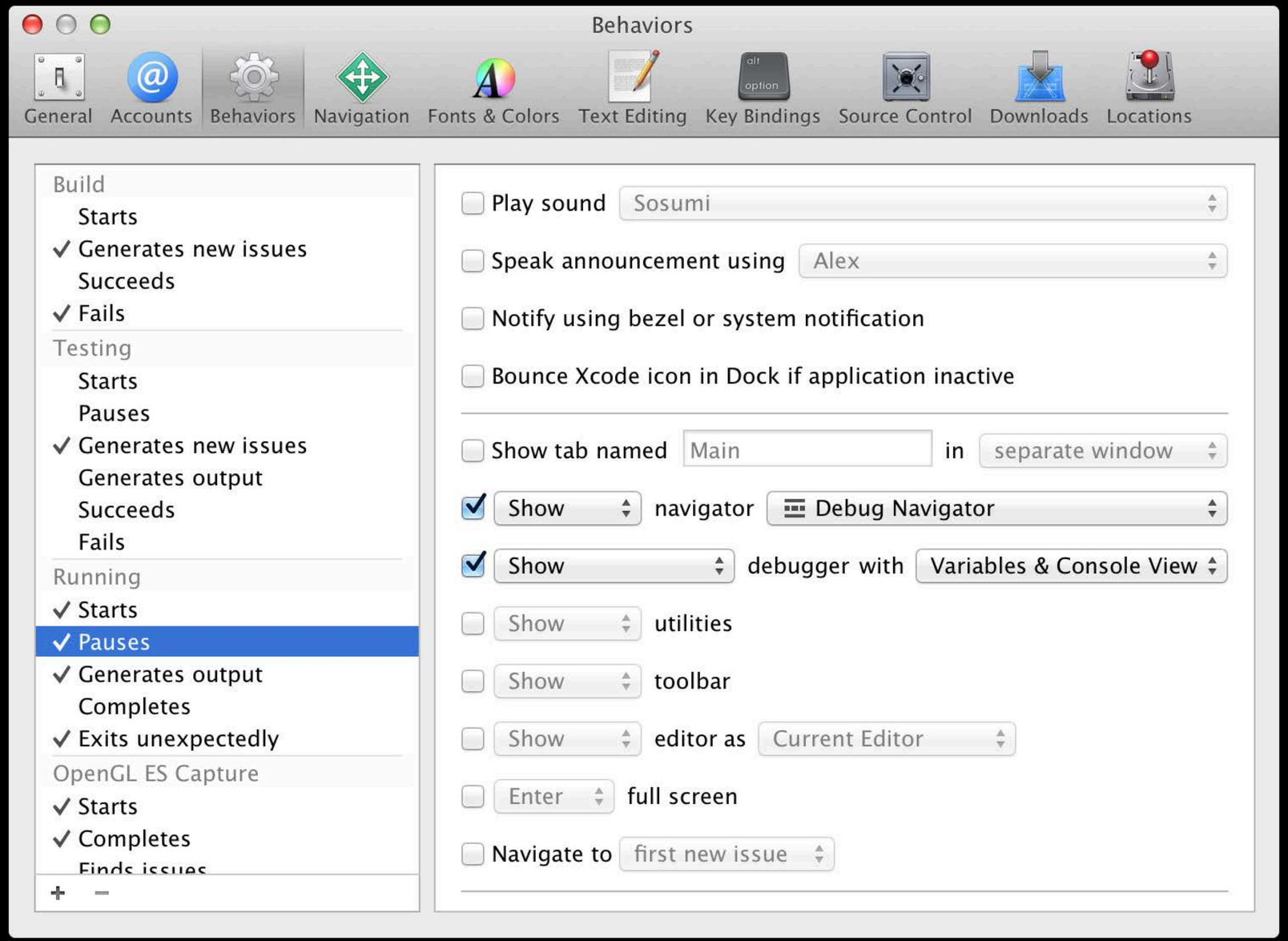


Documentation Window

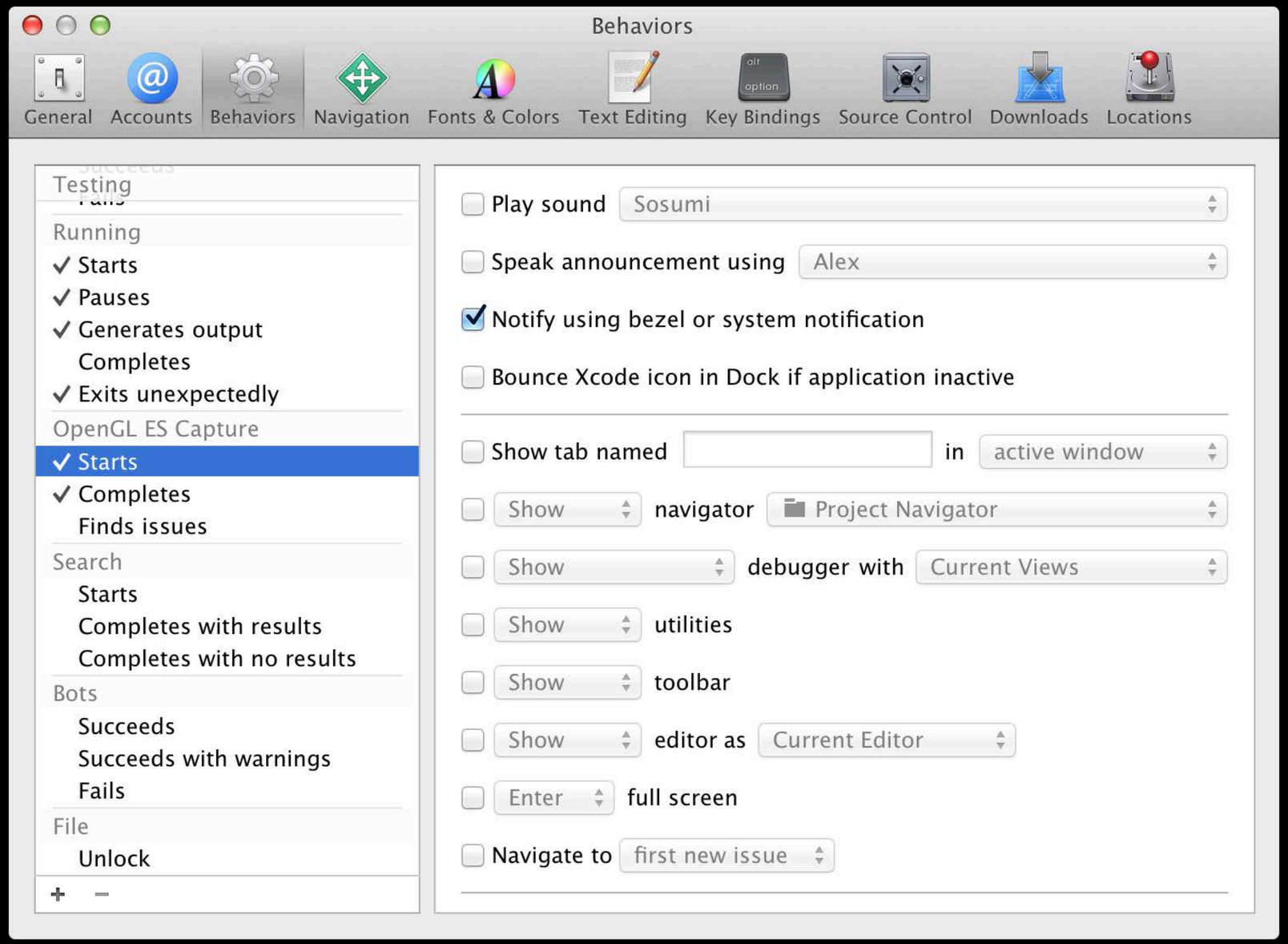


Customizing Xcode's Behaviors

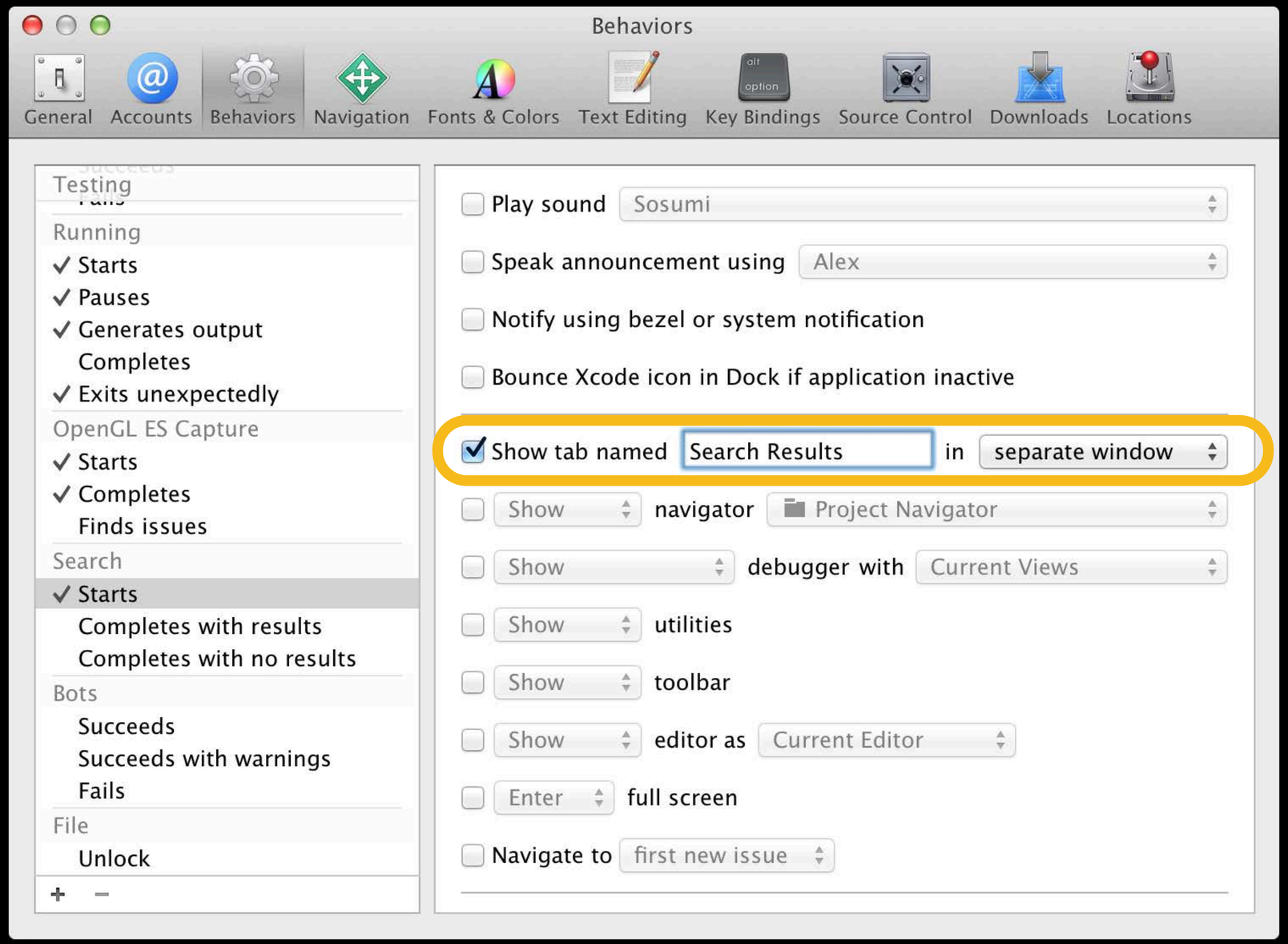
Behaviors


Behaviors
General
Accounts
Behaviors
Navigation
Fonts & Colors
Text Editing
Key Bindings
Source Control
Downloads
Locations
Build
Starts
Generates new issues
Succeeds
Fails
Testing
Starts
Pauses
Generates new issues
Generates output
Succeeds
Fails
Running
Starts
Pauses
Generates output
Completes
Exits unexpectedly
OpenGL ES Capture
Starts
Completes
Play sound
Sosumi
Speak announcement using
Alex
Notify using bezel or system notification
Bounce Xcode icon in Dock if application inactive
Show tab named
Main
in
separate window
Show
navigator
Debug Navigator
Show
debugger with
Variables & Console View
Show
utilities
Show
toolbar
Show
editor as
Current Editor
Enter
full screen
Navigate to
first new issue

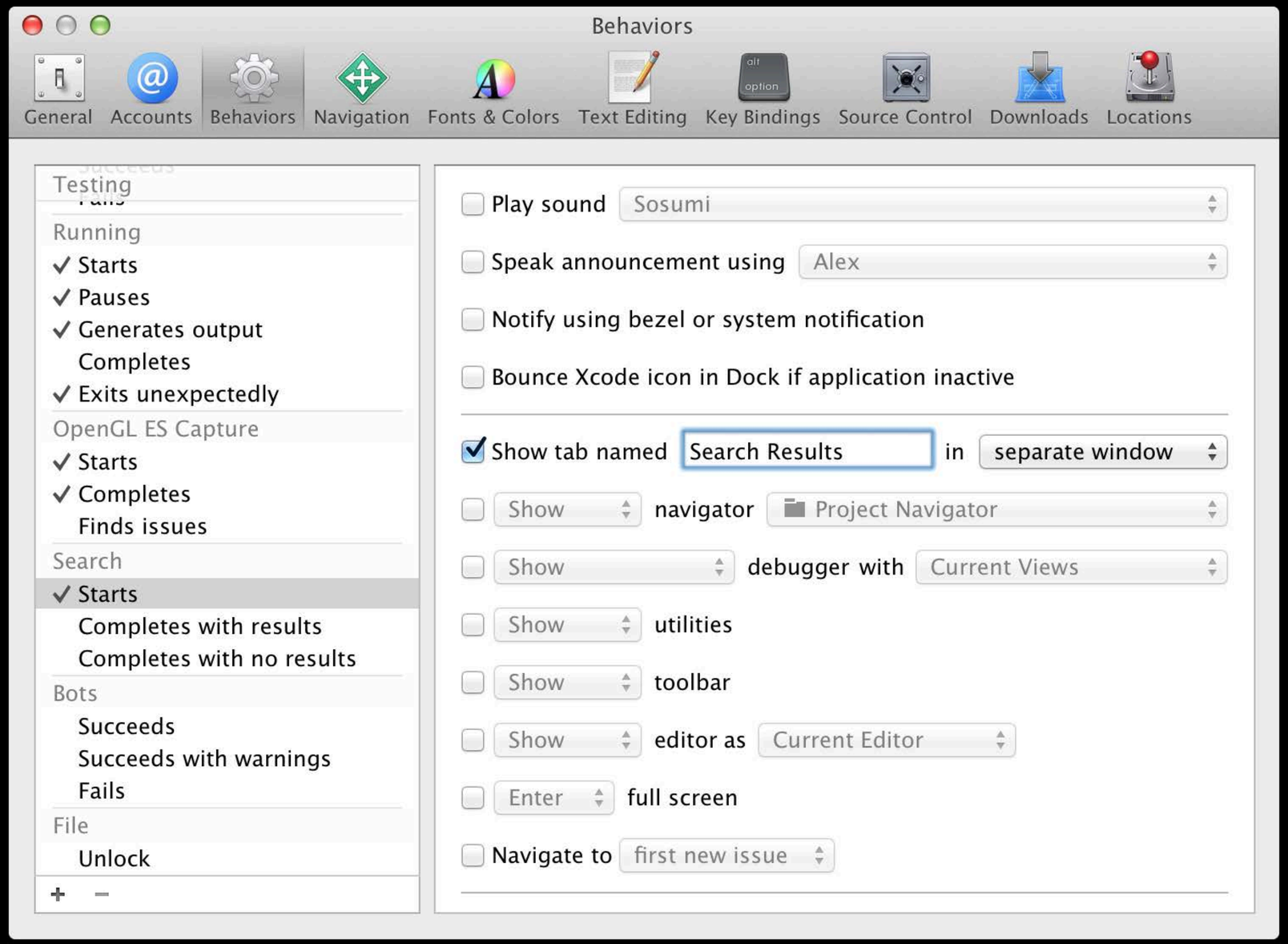
Behaviors


Behaviors
General
Accounts
Behaviors
Navigation
Fonts & Colors
Text Editing
Key Bindings
Source Control
Downloads
Locations
Testing
Running
Starts
Pauses
Generates output
Completes
Exits unexpectedly
OpenGL ES Capture
Starts
Completes
Finds issues
Search
Starts
Completes with results
Completes with no results
Bots
Succeeds
Succeeds with warnings
Fails
File
Unlock
Play sound
Sosumi
Speak announcement using
Alex
Notify using bezel or system notification
Bounce Xcode icon in Dock if application inactive
Show tab named
in
active window
Show
navigator
Project Navigator
Show
debugger with
Current Views
Show
utilities
Show
toolbar
Show
editor as
Current Editor
Enter
full screen
Navigate to
first new issue

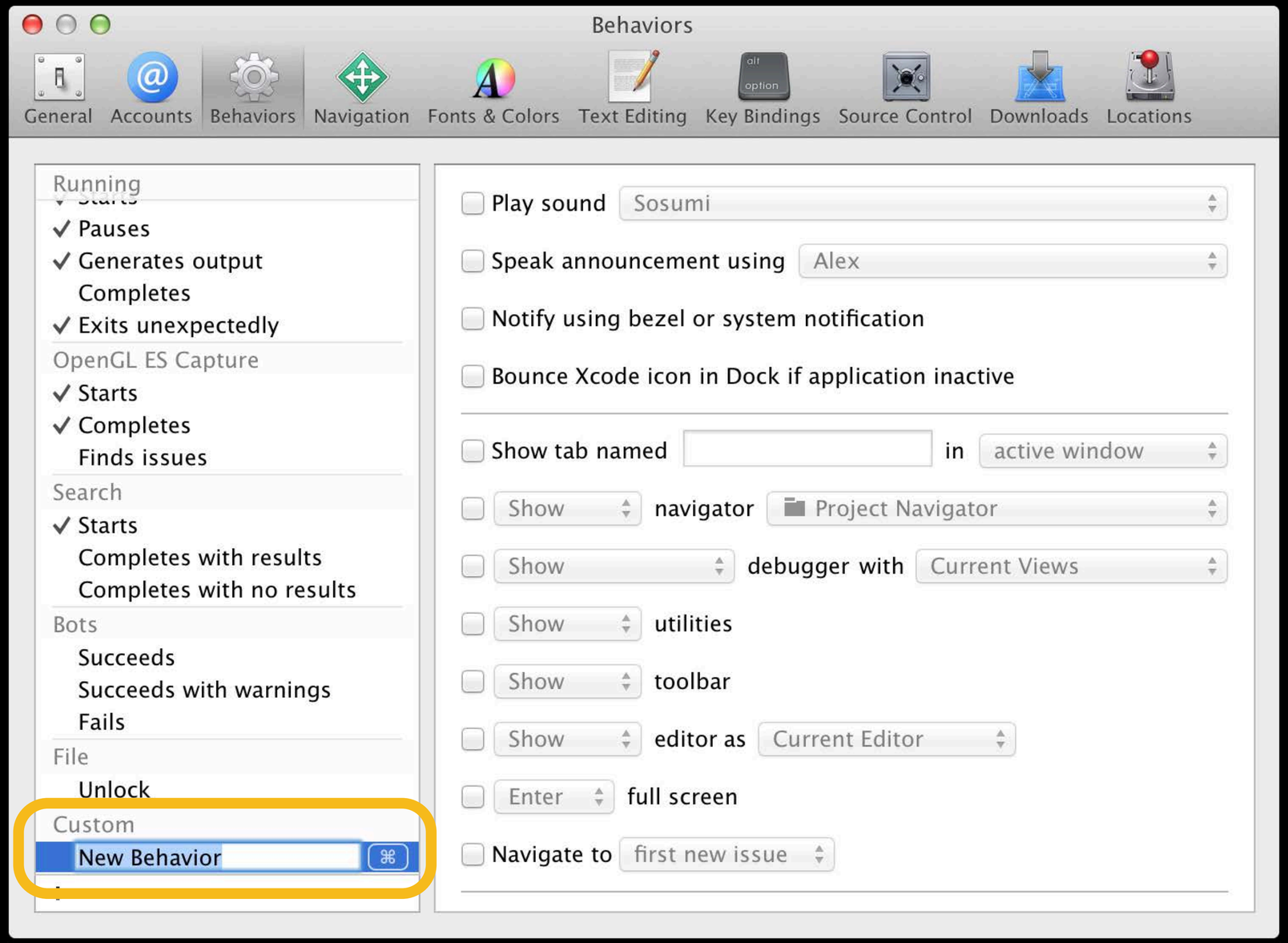
Behaviors


Behaviors
General
Accounts
Behaviors
Navigation
Fonts & Colors
Text Editing
Key Bindings
Source Control
Downloads
Locations
Testing
Running
✓ Starts
✓ Pauses
✓ Generates output
Completes
✓ Exits unexpectedly
OpenGL ES Capture
✓ Starts
✓ Completes
Finds issues
Search
✓ Starts
Completes with results
Completes with no results
Bots
Succeeds
Succeeds with warnings
Fails
File
Unlock
Play sound
Sosumi
Speak announcement using
Alex
Notify using bezel or system notification
Bounce Xcode icon in Dock if application inactive
Show tab named
Search Results
in
separate window
Show
navigator
Project Navigator
Show
debugger with
Current Views
Show
utilities
Show
toolbar
Show
editor as
Current Editor
Enter
full screen
Navigate to
first new issue

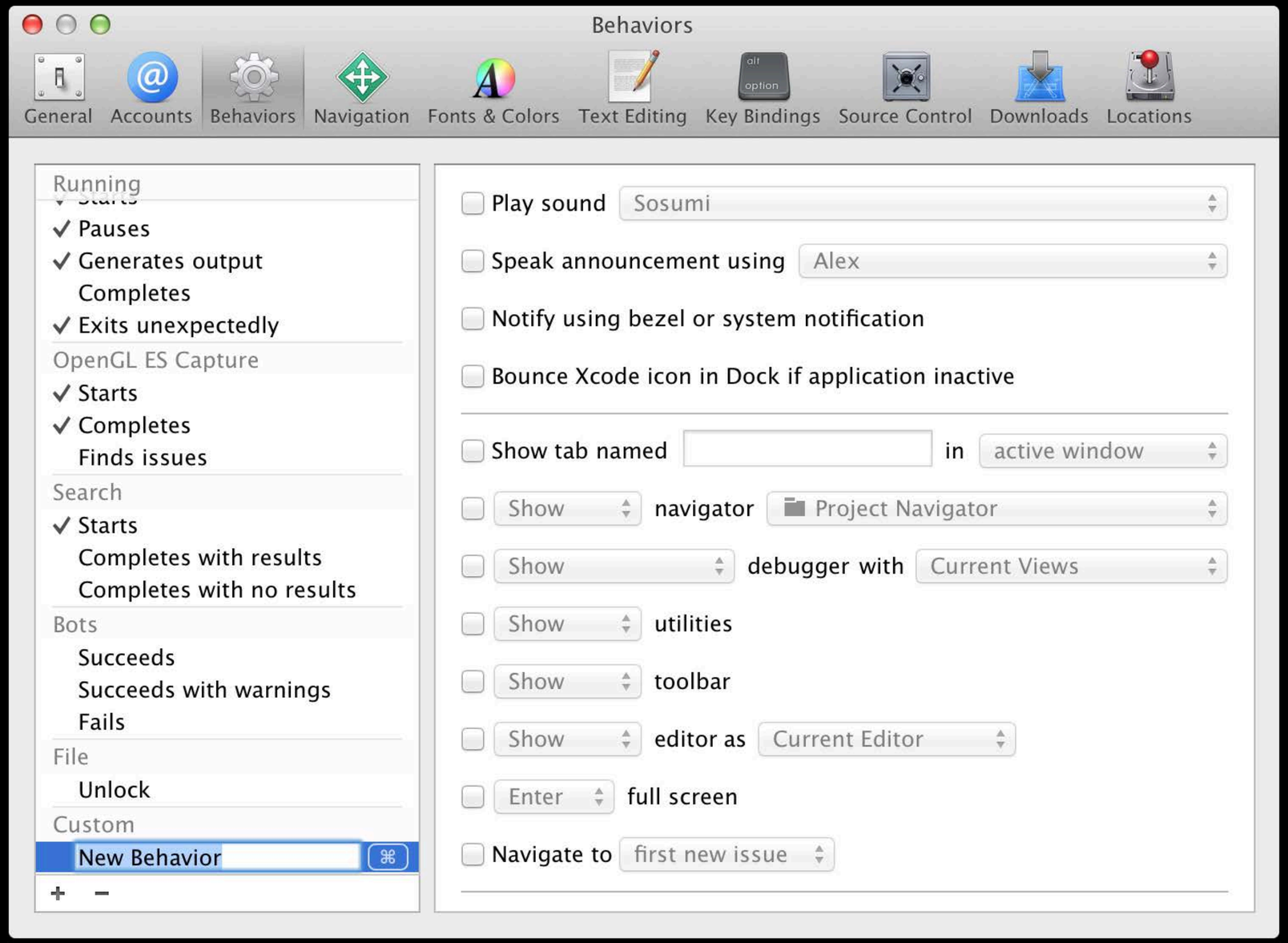
Behaviors


Behaviors
General
Accounts
Behaviors
Navigation
Fonts & Colors
Text Editing
Key Bindings
Source Control
Downloads
Locations
Testing
Running
✓ Starts
✓ Pauses
✓ Generates output
Completes
✓ Exits unexpectedly
OpenGL ES Capture
✓ Starts
✓ Completes
Finds issues
Search
✓ Starts
Completes with results
Completes with no results
Bots
Succeeds
Succeeds with warnings
Fails
File
Unlock
Play sound
Sosumi
Speak announcement using
Alex
Notify using bezel or system notification
Bounce Xcode icon in Dock if application inactive
Show tab named
Search Results
in
separate window
Show
navigator
Project Navigator
Show
debugger with
Current Views
Show
utilities
Show
toolbar
Show
editor as
Current Editor
Enter
full screen
Navigate to
first new issue

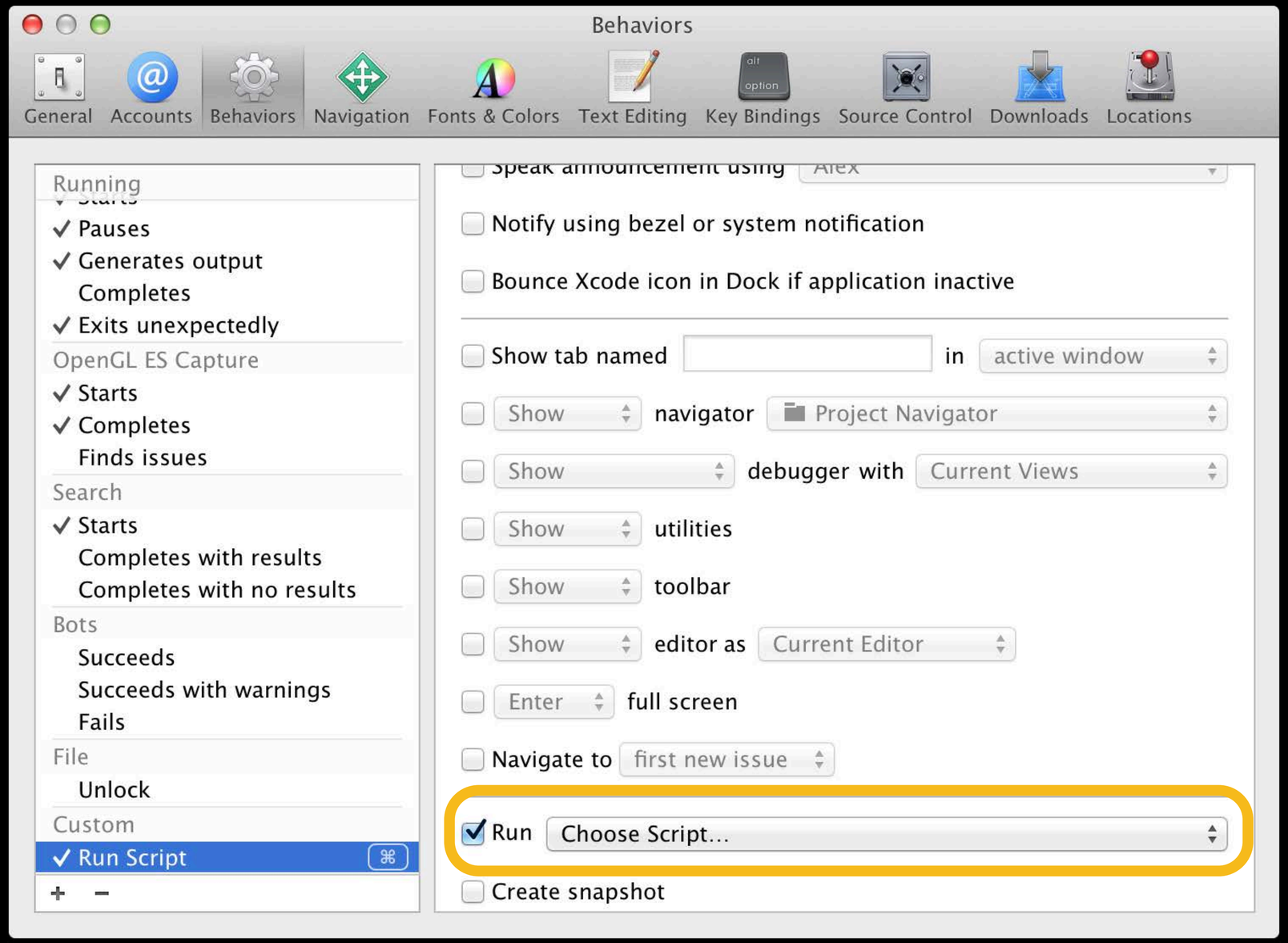
Behaviors


Behaviors
General
Accounts
Behaviors
Navigation
Fonts & Colors
Text Editing
Key Bindings
Source Control
Downloads
Locations
Running
Pauses
Generates output
Completes
Exits unexpectedly
OpenGL ES Capture
Starts
Completes
Finds issues
Search
Starts
Completes with results
Completes with no results
Bots
Succeeds
Succeeds with warnings
Fails
File
Unlock
Custom
New Behavior
Play sound
Sosumi
Speak announcement using
Alex
Notify using bezel or system notification
Bounce Xcode icon in Dock if application inactive
Show tab named
in
active window
Show
navigator
Project Navigator
Show
debugger with
Current Views
Show
utilities
Show
toolbar
Show
editor as
Current Editor
Enter
full screen
Navigate to
first new issue

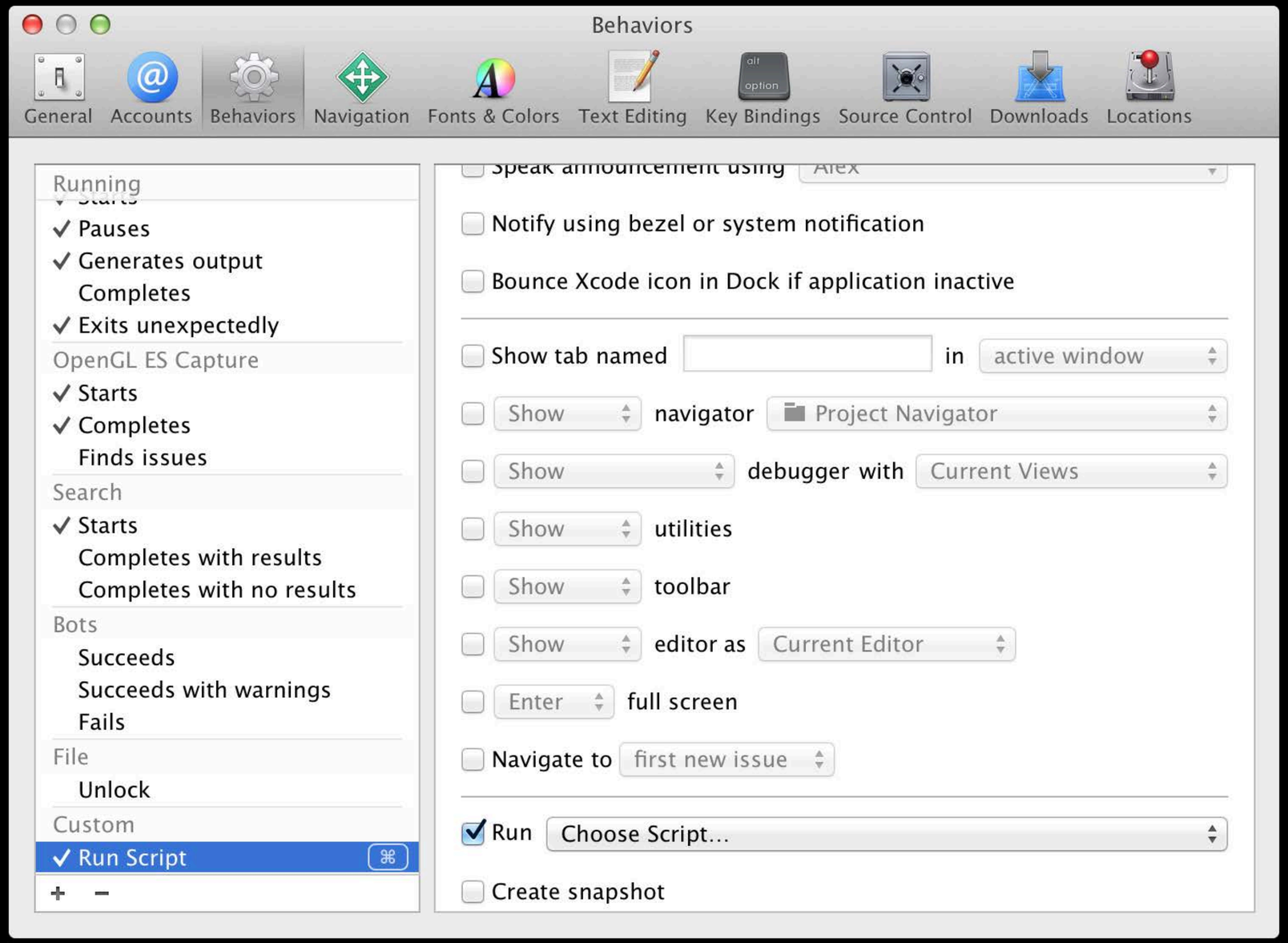
Behaviors


Behaviors
General
Accounts
Behaviors
Navigation
Fonts & Colors
Text Editing
Key Bindings
Source Control
Downloads
Locations
Running
Pauses
Generates output
Completes
Exits unexpectedly
OpenGL ES Capture
Starts
Completes
Finds issues
Search
Starts
Completes with results
Completes with no results
Bots
Succeeds
Succeeds with warnings
Fails
File
Unlock
Custom
New Behavior
Play sound
Sosumi
Speak announcement using
Alex
Notify using bezel or system notification
Bounce Xcode icon in Dock if application inactive
Show tab named
in
active window
Show
navigator
Project Navigator
Show
debugger with
Current Views
Show
utilities
Show
toolbar
Show
editor as
Current Editor
Enter
full screen
Navigate to
first new issue

Behaviors


Behaviors
General
Accounts
Behaviors
Navigation
Fonts & Colors
Text Editing
Key Bindings
Source Control
Downloads
Locations
Running
Pauses
Generates output
Completes
Exits unexpectedly
OpenGL ES Capture
Starts
Completes
Finds issues
Search
Starts
Completes with results
Completes with no results
Bots
Succeeds
Succeeds with warnings
Fails
File
Unlock
Custom
Run Script
Speak announcement using
Alex
Notify using bezel or system notification
Bounce Xcode icon in Dock if application inactive
Show tab named
in
active window
Show
navigator
Project Navigator
Show
debugger with
Current Views
Show
utilities
Show
toolbar
Show
editor as
Current Editor
Enter
full screen
Navigate to
first new issue
Run
Choose Script...
Create snapshot

Behaviors


Behaviors
General
Accounts
Behaviors
Navigation
Fonts & Colors
Text Editing
Key Bindings
Source Control
Downloads
Locations
Running
Pauses
Generates output
Completes
Exits unexpectedly
OpenGL ES Capture
Starts
Completes
Finds issues
Search
Starts
Completes with results
Completes with no results
Bots
Succeeds
Succeeds with warnings
Fails
File
Unlock
Custom
Run Script
Speak announcement using
Alex
Notify using bezel or system notification
Bounce Xcode icon in Dock if application inactive
Show tab named
in
active window
Show
navigator
Project Navigator
Show
debugger with
Current Views
Show
utilities
Show
toolbar
Show
editor as
Current Editor
Enter
full screen
Navigate to
first new issue
Run
Choose Script...
Create snapshot

Fundamental Xcode Concepts

What You'll Learn to Do

What You'll Learn to Do

- Set up a workspace

What You'll Learn to Do

- Set up a workspace
- Configure settings

What You'll Learn to Do

- Set up a workspace
- Configure settings
- Run, test, profile, analyze, and archive

What You'll Learn to Do

- Set up a workspace
- Configure settings
- Run, test, profile, analyze, and archive
- Understand Xcode's model

Fundamental Concepts

Fundamental Concepts

- Workspaces

Fundamental Concepts

- Workspaces
- Projects
  - Build configurations

Fundamental Concepts

- Workspaces
- Projects
  - Build configurations
- Targets
  - Build settings
  - Build phases

Fundamental Concepts

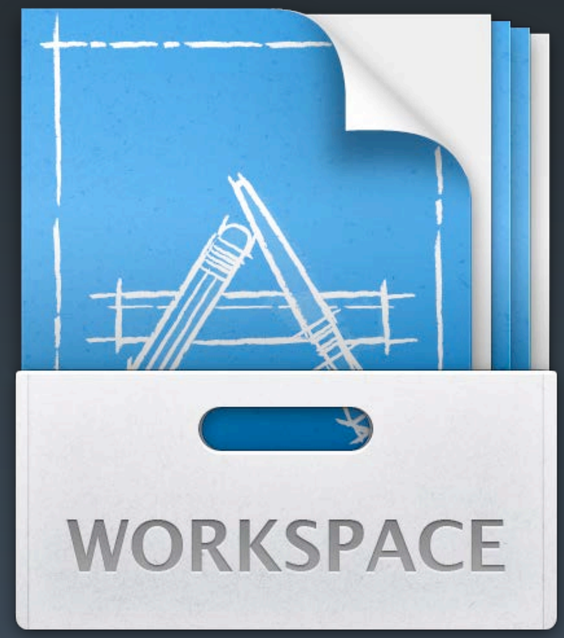
- Workspaces
- Projects
  - Build configurations
- Targets
  - Build settings
  - Build phases
- Schemes

Fundamental Concepts

- Workspaces
- Projects
  - Build configurations
- Targets
  - Build settings
  - Build phases
- Schemes
- Run destinations

Workspaces

Collections of files and projects with state



What is a Workspace?

What is a Workspace?

- Contains references to projects and other files

What is a Workspace?

- Contains references to projects and other files
- Saves state (open windows, breakpoints, etc.)

What is a Workspace?

- Contains references to projects and other files
- Saves state (open windows, breakpoints, etc.)
- Provides a unique location for build products

What is a Workspace?

- Contains references to projects and other files
- Saves state (open windows, breakpoints, etc.)
- Provides a unique location for build products
- Provides a symbol index

Why Use a Workspace?

Why Use a Workspace?

- Groups together projects you want to use together

Why Use a Workspace?

- Groups together projects you want to use together
- Finds implicit dependencies between targets

Why Use a Workspace?

- Groups together projects you want to use together
- Finds implicit dependencies between targets
- Opening a project gives you an implicit workspace

Projects

Collections of files, targets, schemes, and build configurations

What's in a Project?

What's in a Project?

- References to source files

What's in a Project?

- References to source files
- Targets which build products

What's in a Project?

- References to source files
- Targets which build products
- Schemes which build targets and perform actions

What's in a Project?

- References to source files
- Targets which build products
- Schemes which build targets and perform actions
- Build configurations which select variants of target build settings

Build Configurations

Build setting variants

Build Configurations

Build setting variants

Build Configurations

Build setting variants

- Build settings can have per-configuration variants

Build Configurations

Build setting variants

- Build settings can have per-configuration variants
- Debug and Release are the default configurations

Build Configurations

Build setting variants

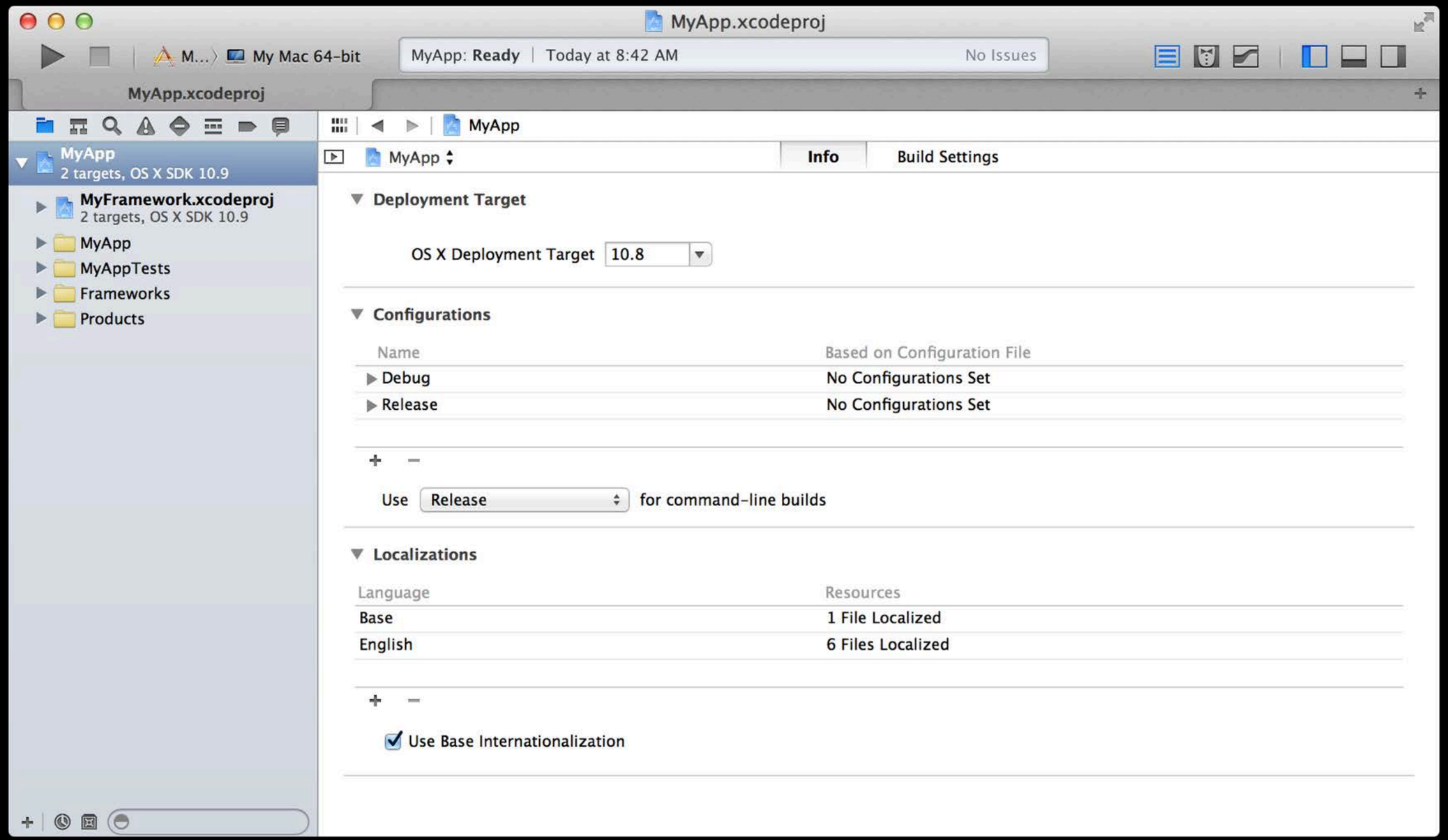
- Build settings can have per-configuration variants
- Debug and Release are the default configurations
- Most settings don’t need variants; a few do

Build Configurations

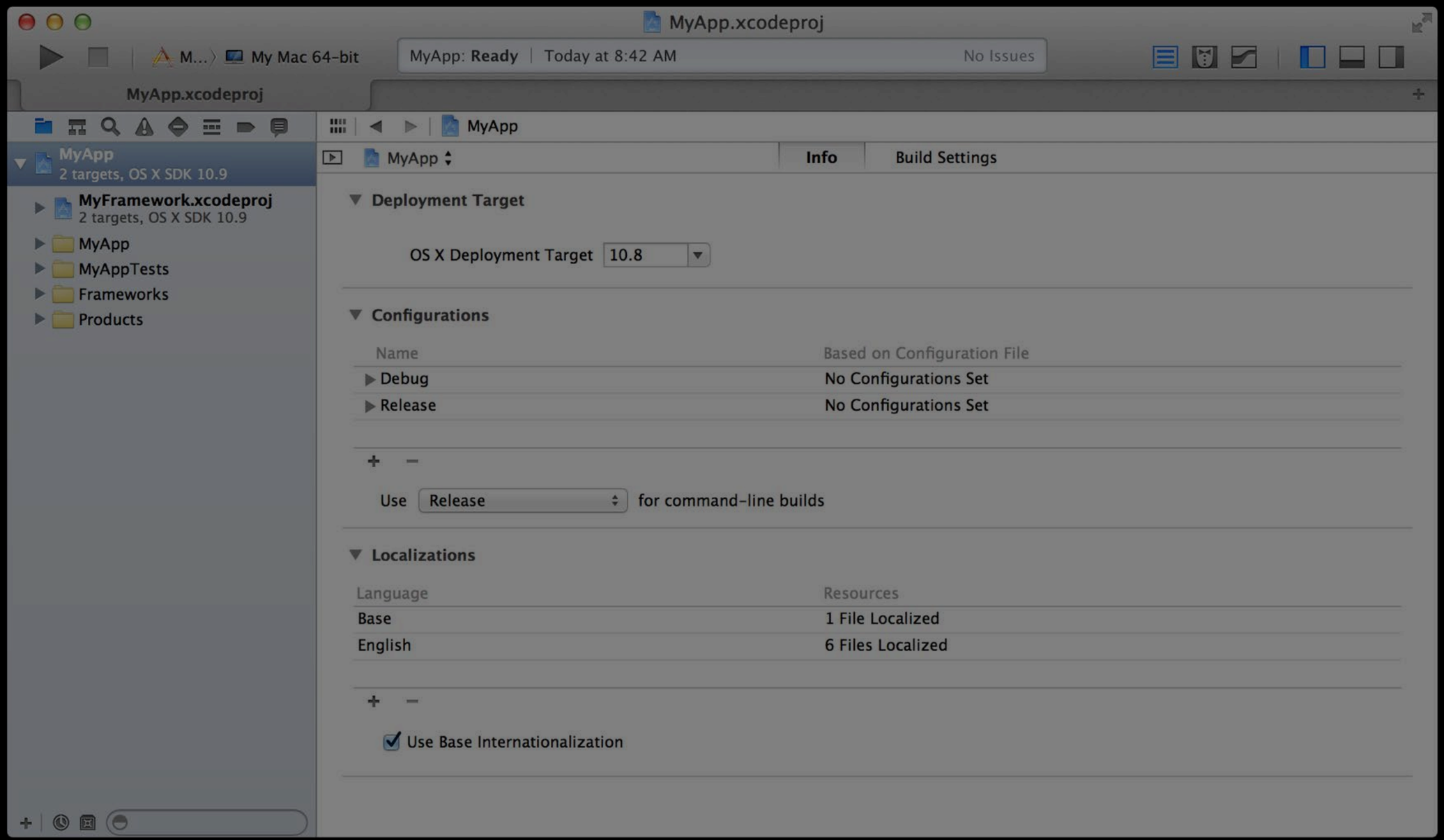
Build setting variants

- Build settings can have per-configuration variants
- Debug and Release are the default configurations
- Most settings don’t need variants; a few do
- You can create additional configurations

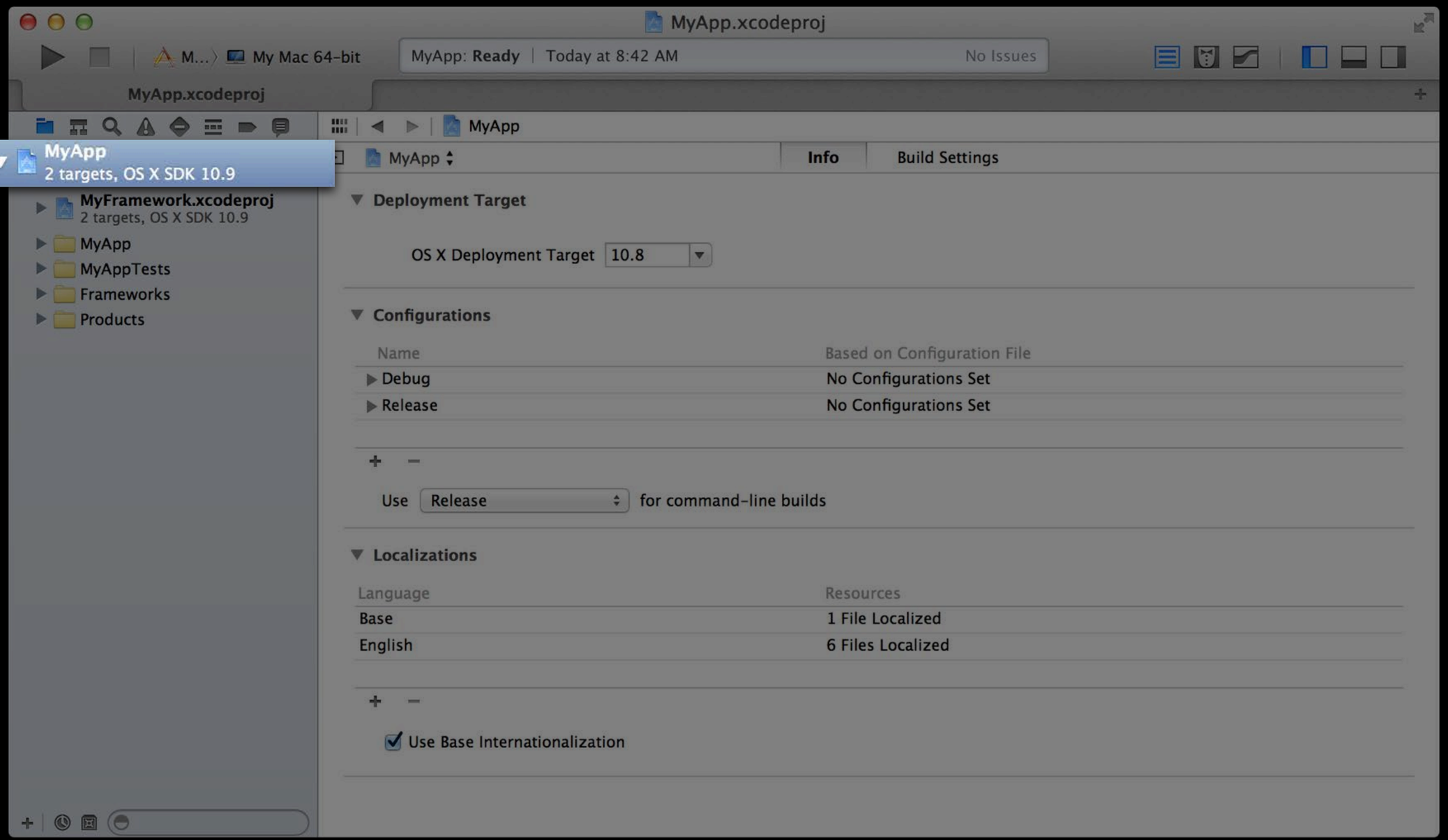
The Project Editor



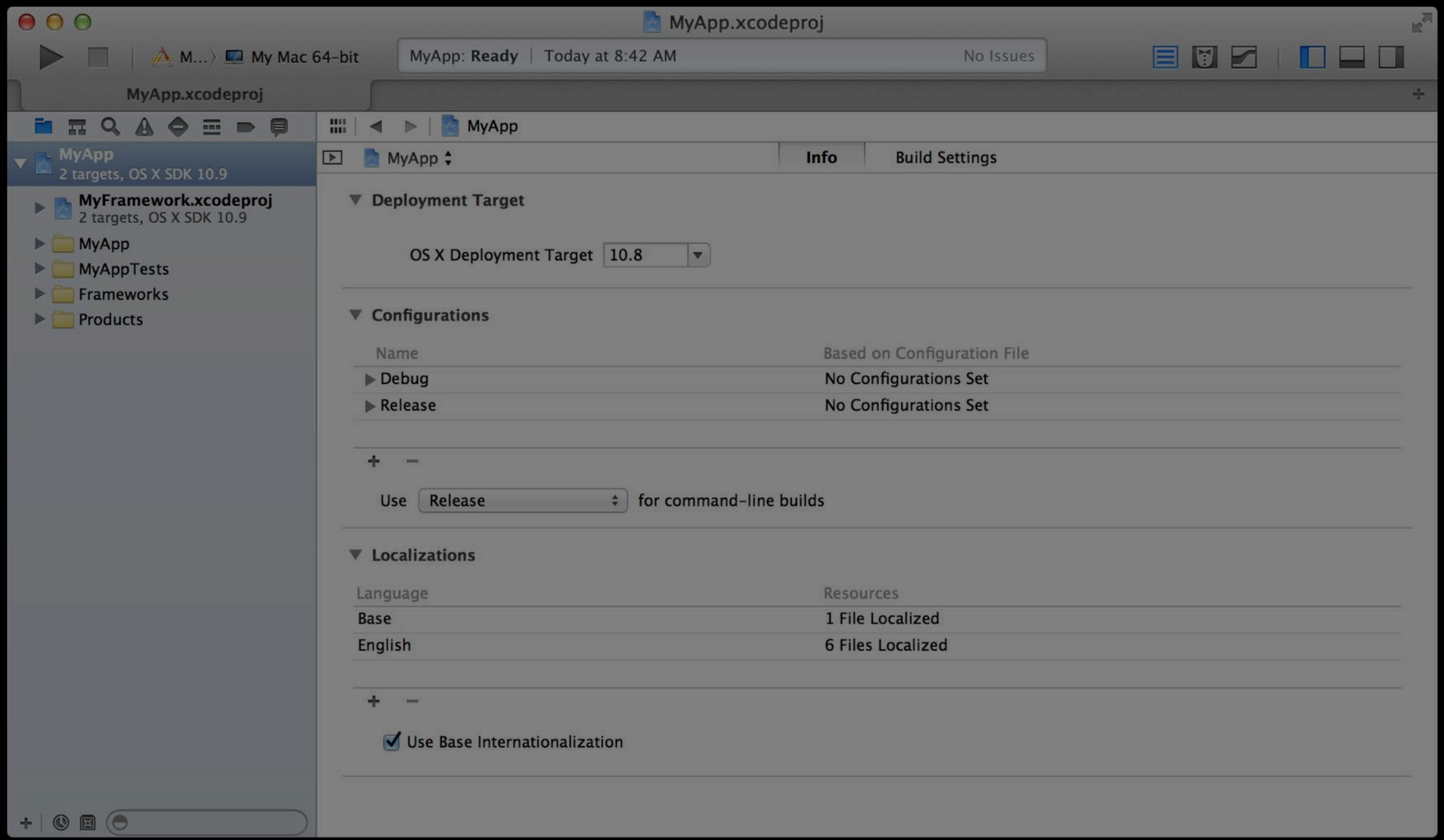
The Project Editor



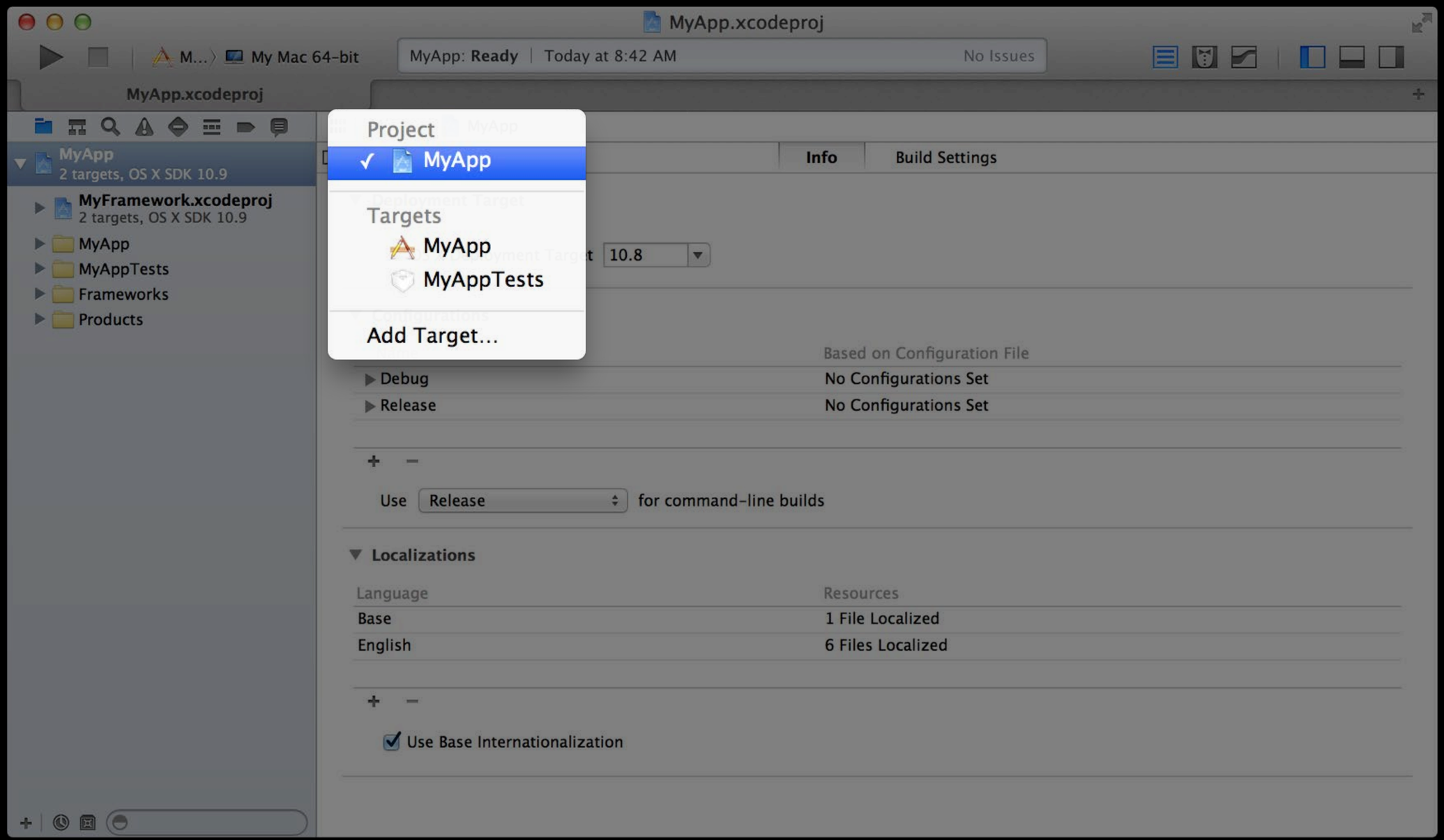
The Project Editor



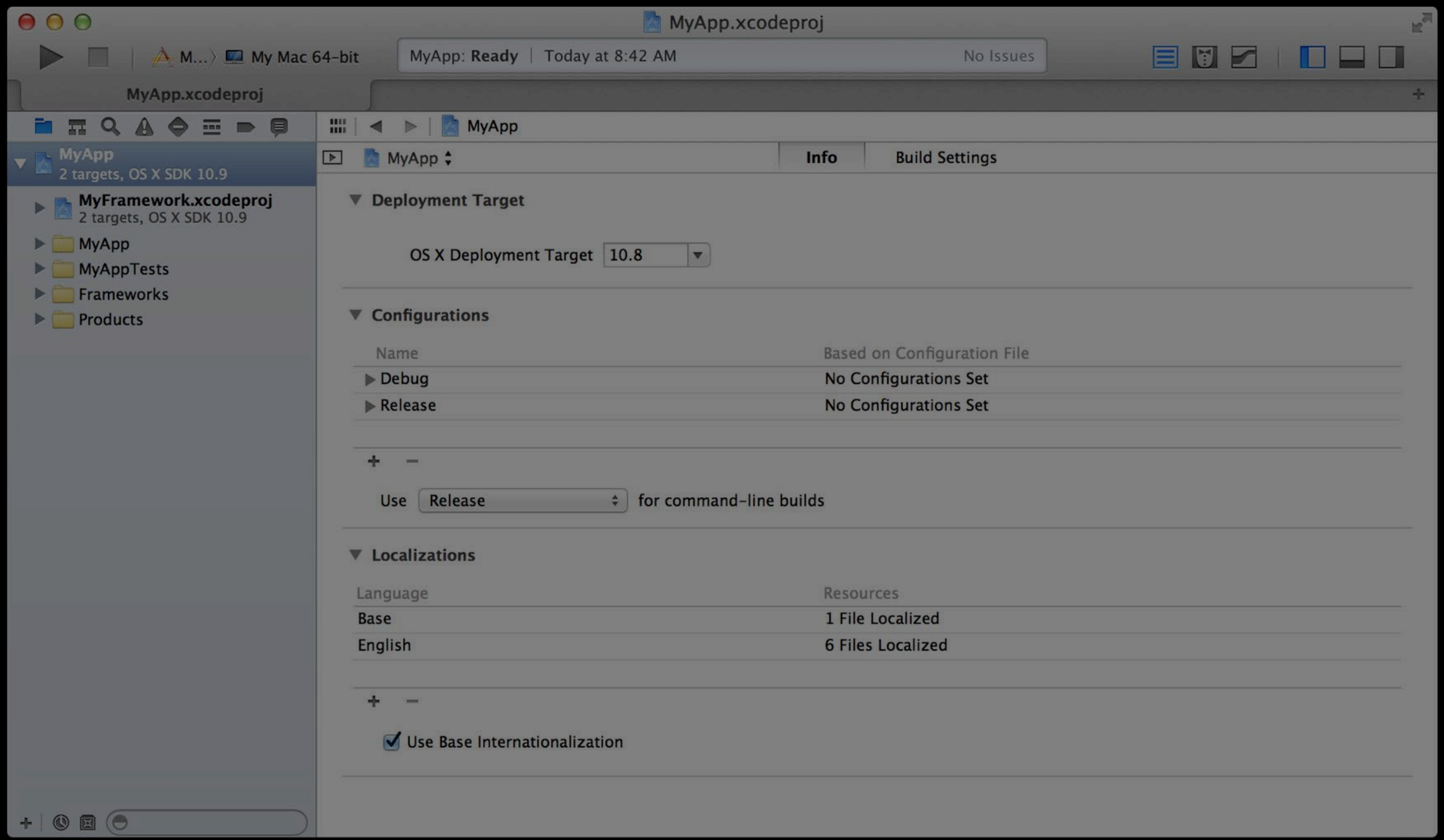
The Project Editor



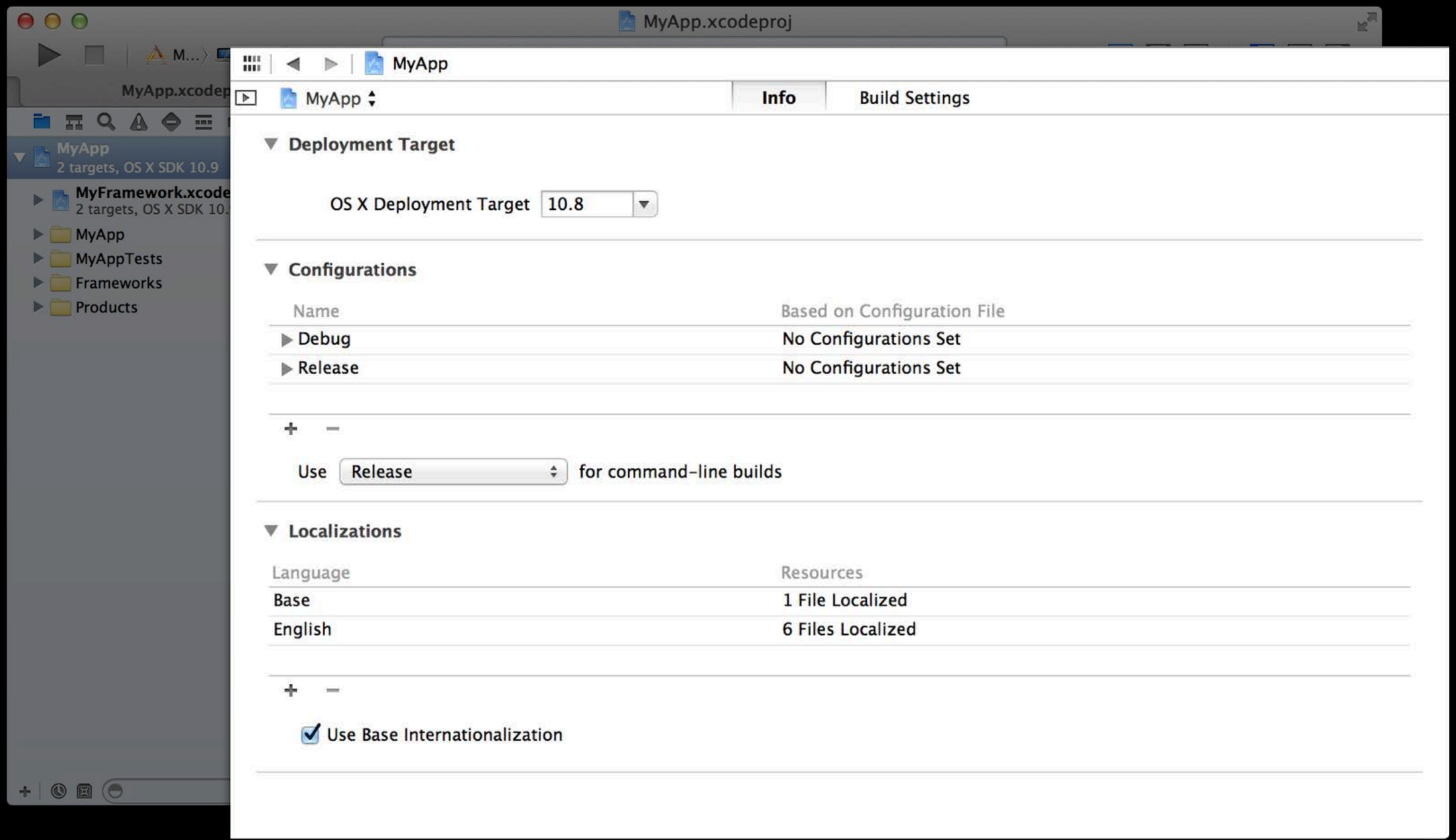
The Project Editor



The Project Editor



The Project Editor

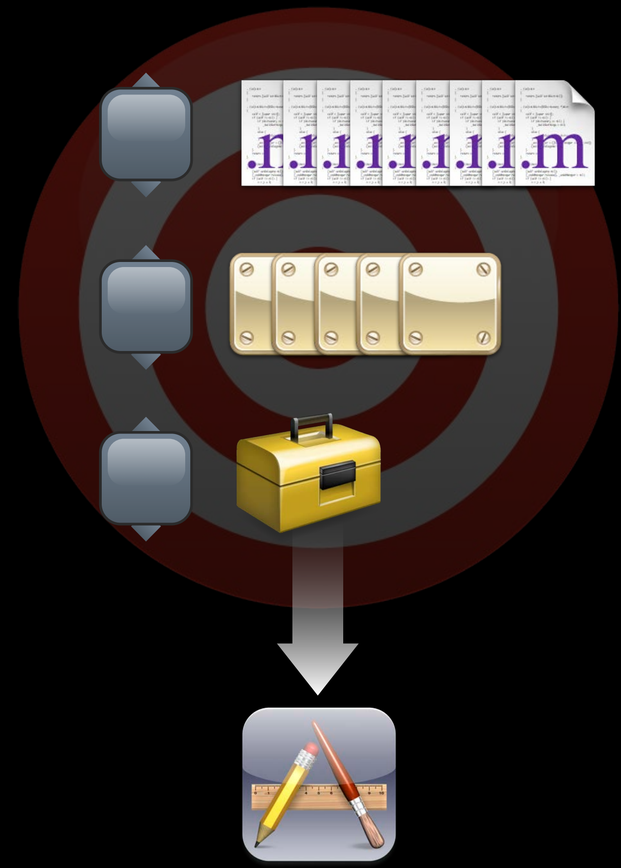


Targets

Instructions for building one product

What's in a Target?

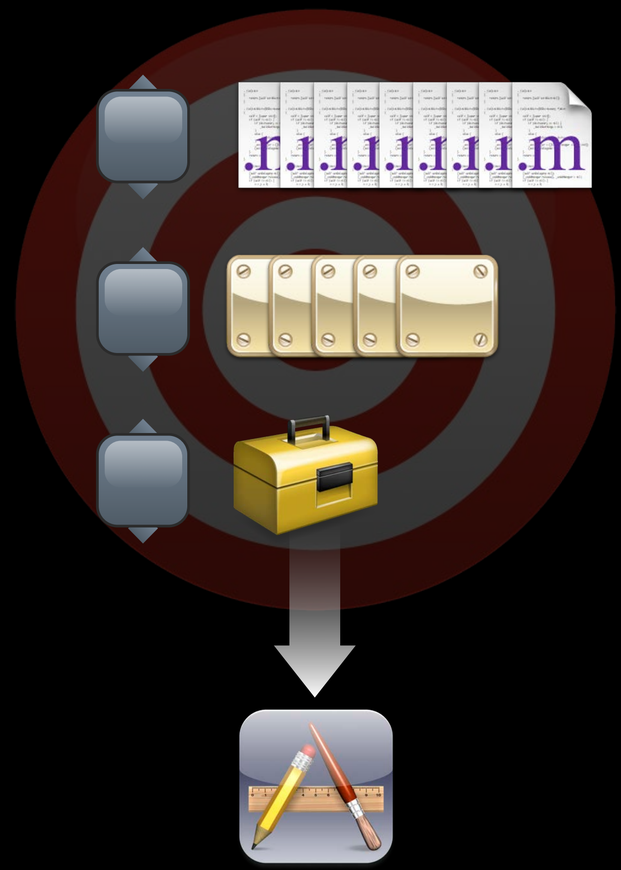
Instructions for building one product



What's in a Target?

Instructions for building one product

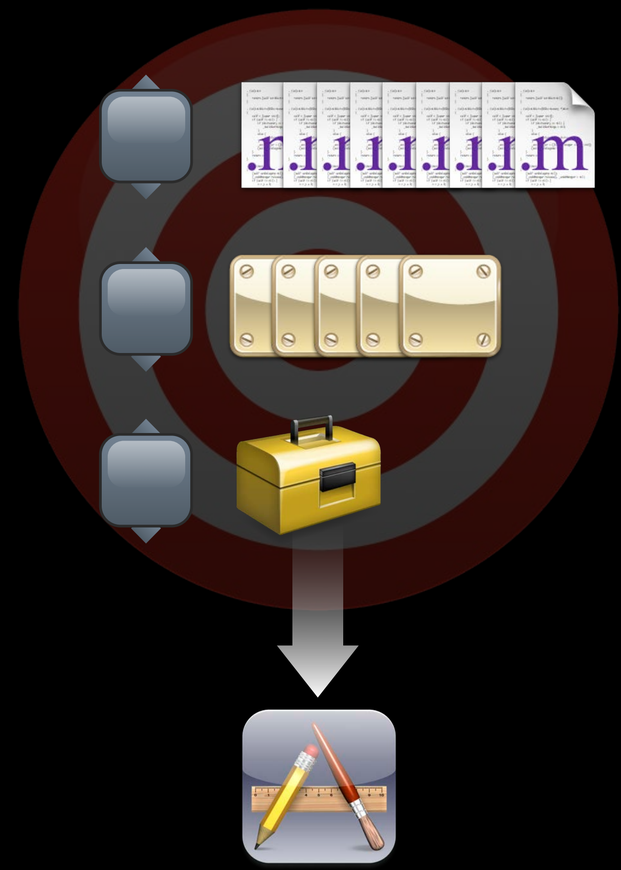
- References some or all source files in the project



What's in a Target?

Instructions for building one product

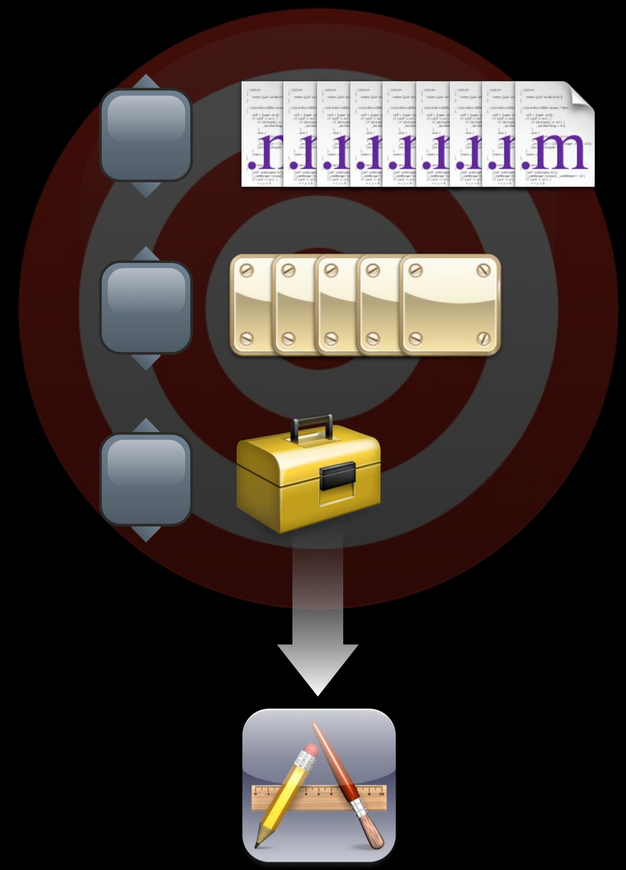
- References some or all source files in the project
- Contains build phases—the high-level sequence of steps



What's in a Target?

Instructions for building one product

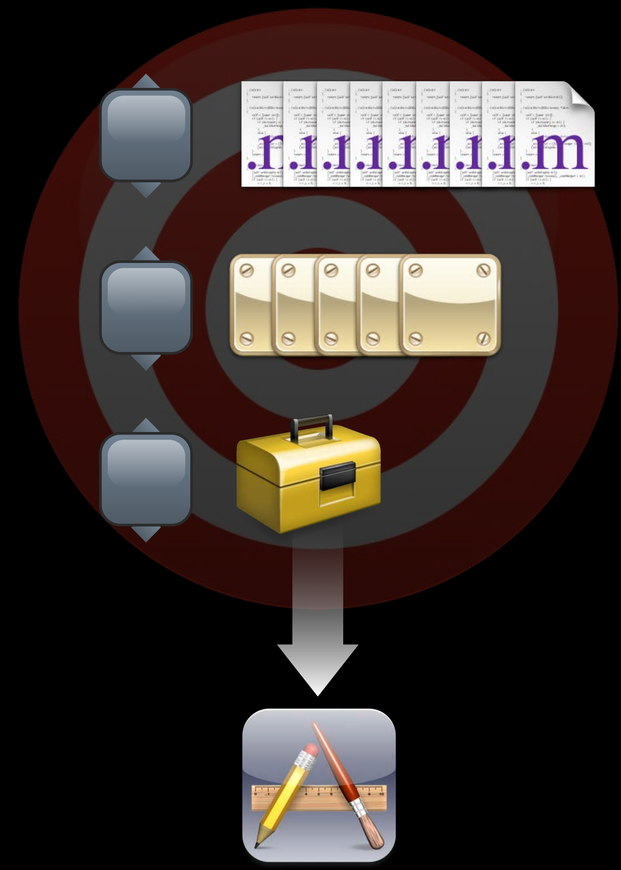
- References some or all source files in the project
- Contains build phases—the high-level sequence of steps
- Build rules determine how to handle each file type



What's in a Target?

Instructions for building one product

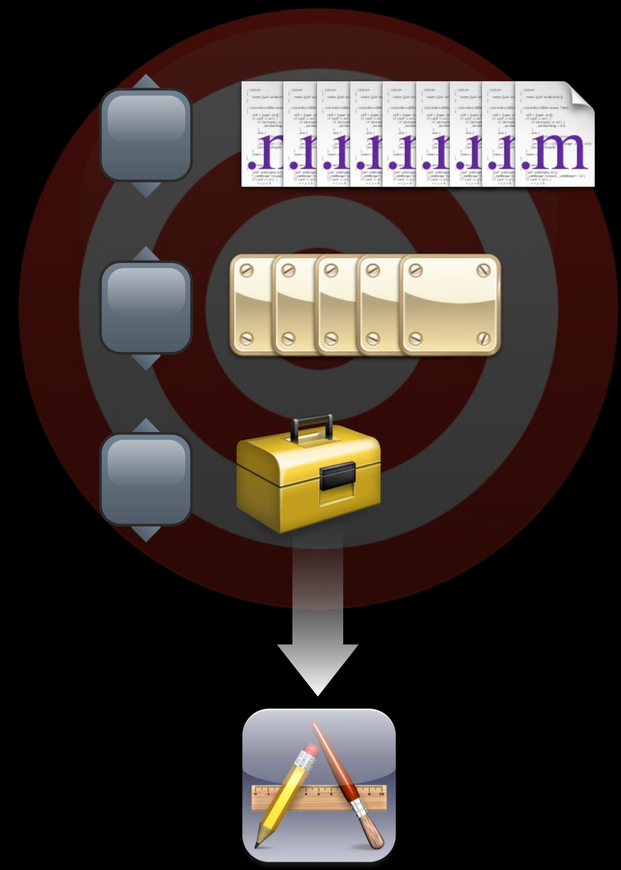
- References some or all source files in the project
- Contains build phases—the high-level sequence of steps
- Build rules determine how to handle each file type
- Build settings control how it is done



What's in a Target?

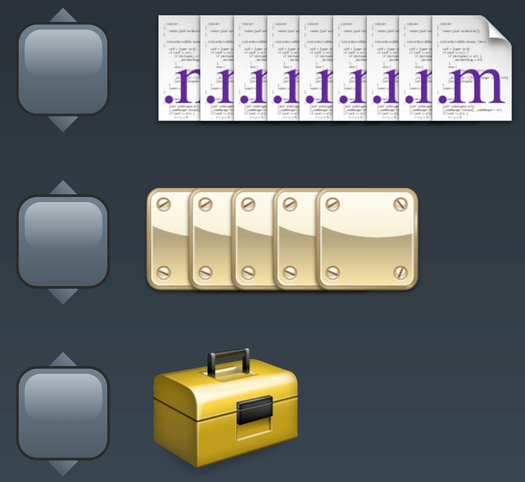
Instructions for building one product

- References some or all source files in the project
- Contains build phases—the high-level sequence of steps
- Build rules determine how to handle each file type
- Build settings control how it is done
- Can depend on one or more other targets

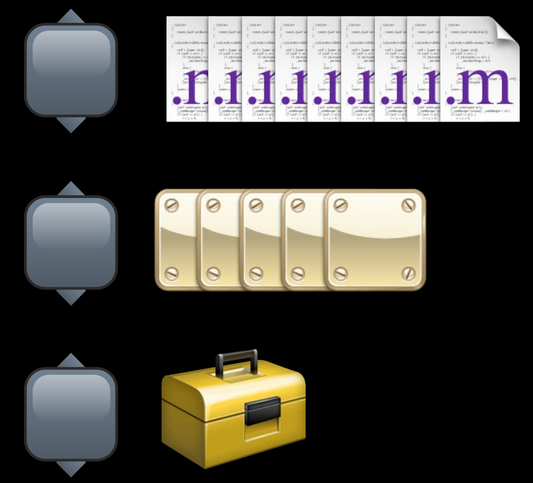


Build Phases

Set of steps taken to build a product

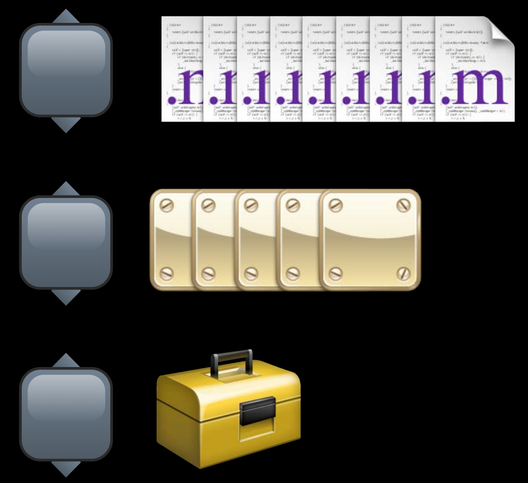


Why Use Build Phases?



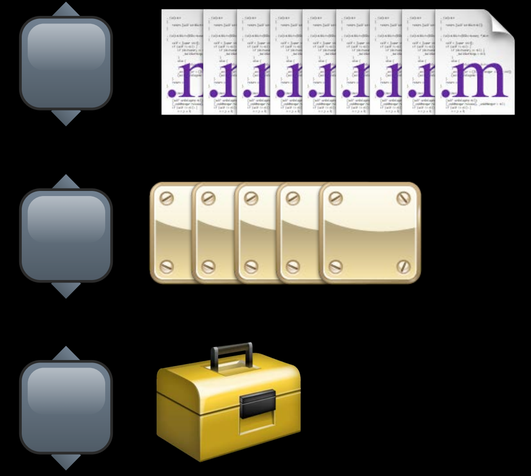
Why Use Build Phases?

- To set per-file compiler flags



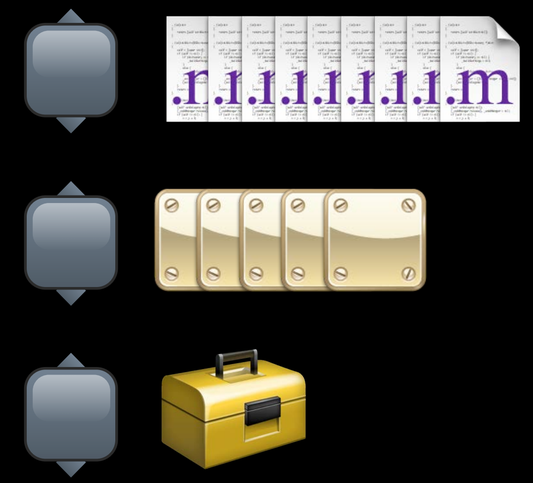
Why Use Build Phases?

- To set per-file compiler flags
- To add resources to your product



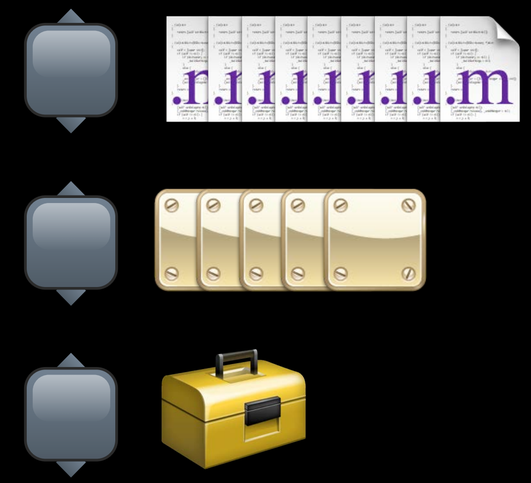
Why Use Build Phases?

- To set per-file compiler flags
- To add resources to your product
- To make a target depend on another target

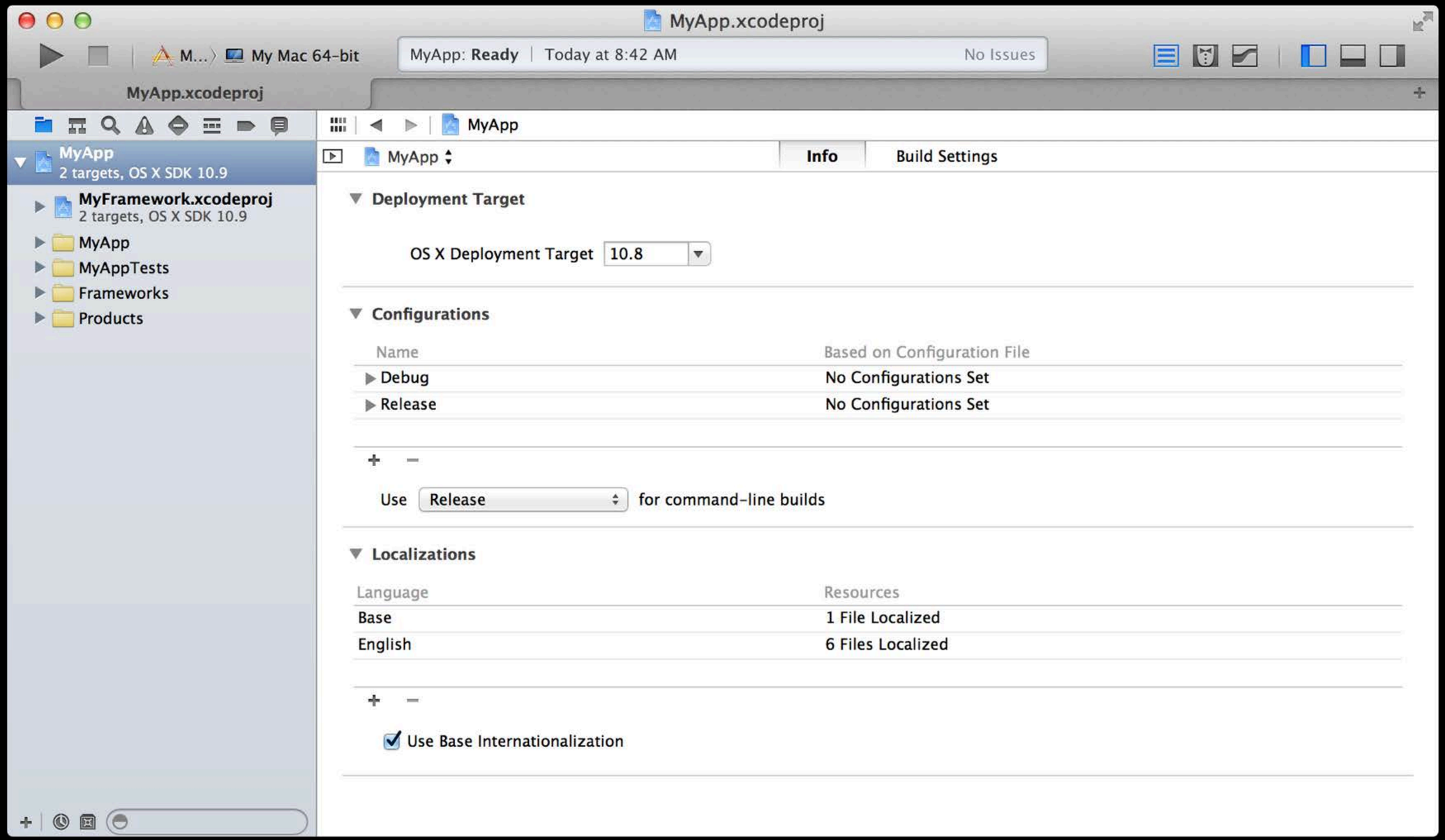


Why Use Build Phases?

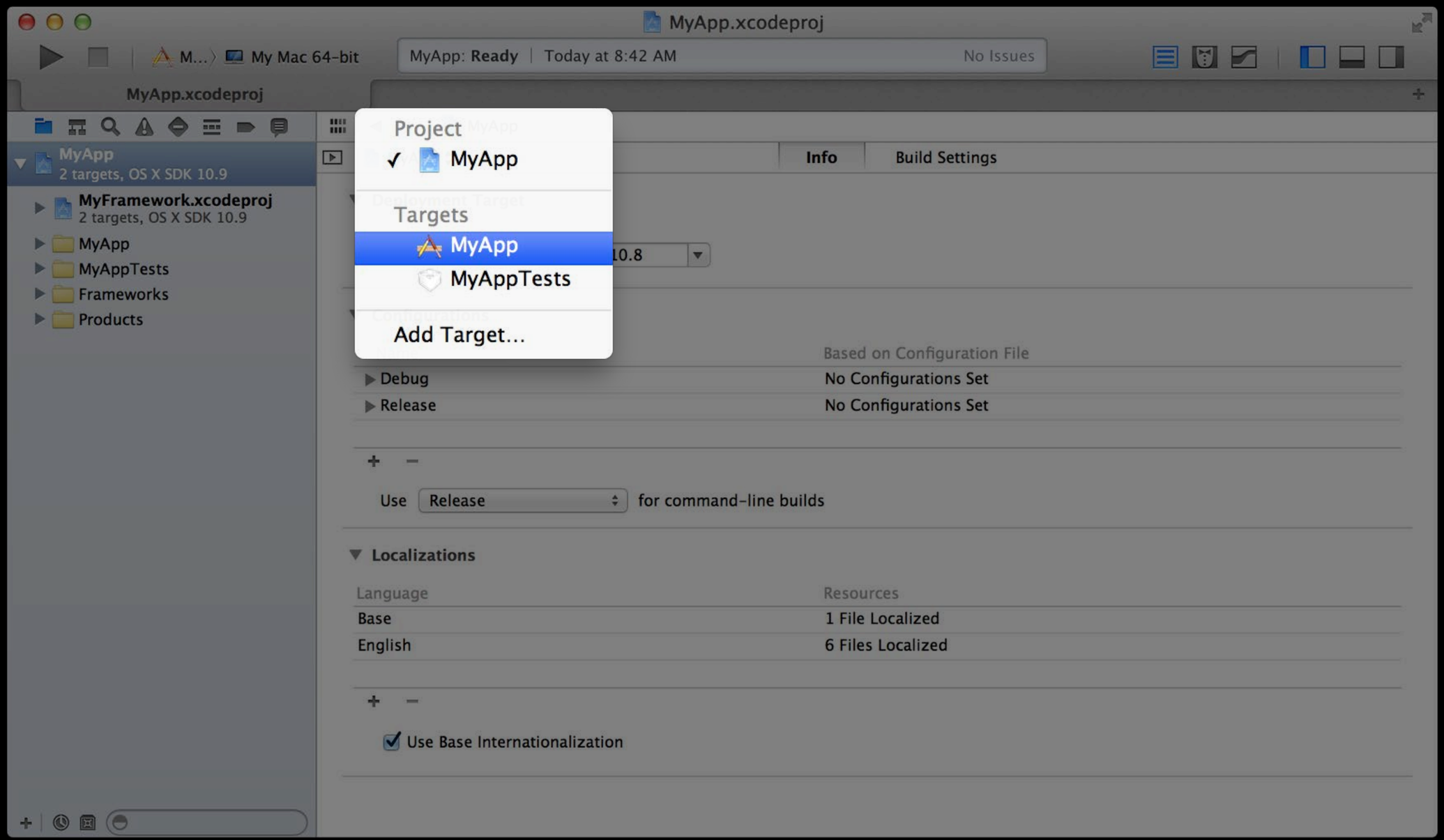
- To set per-file compiler flags
- To add resources to your product
- To make a target depend on another target
- To run a script while building



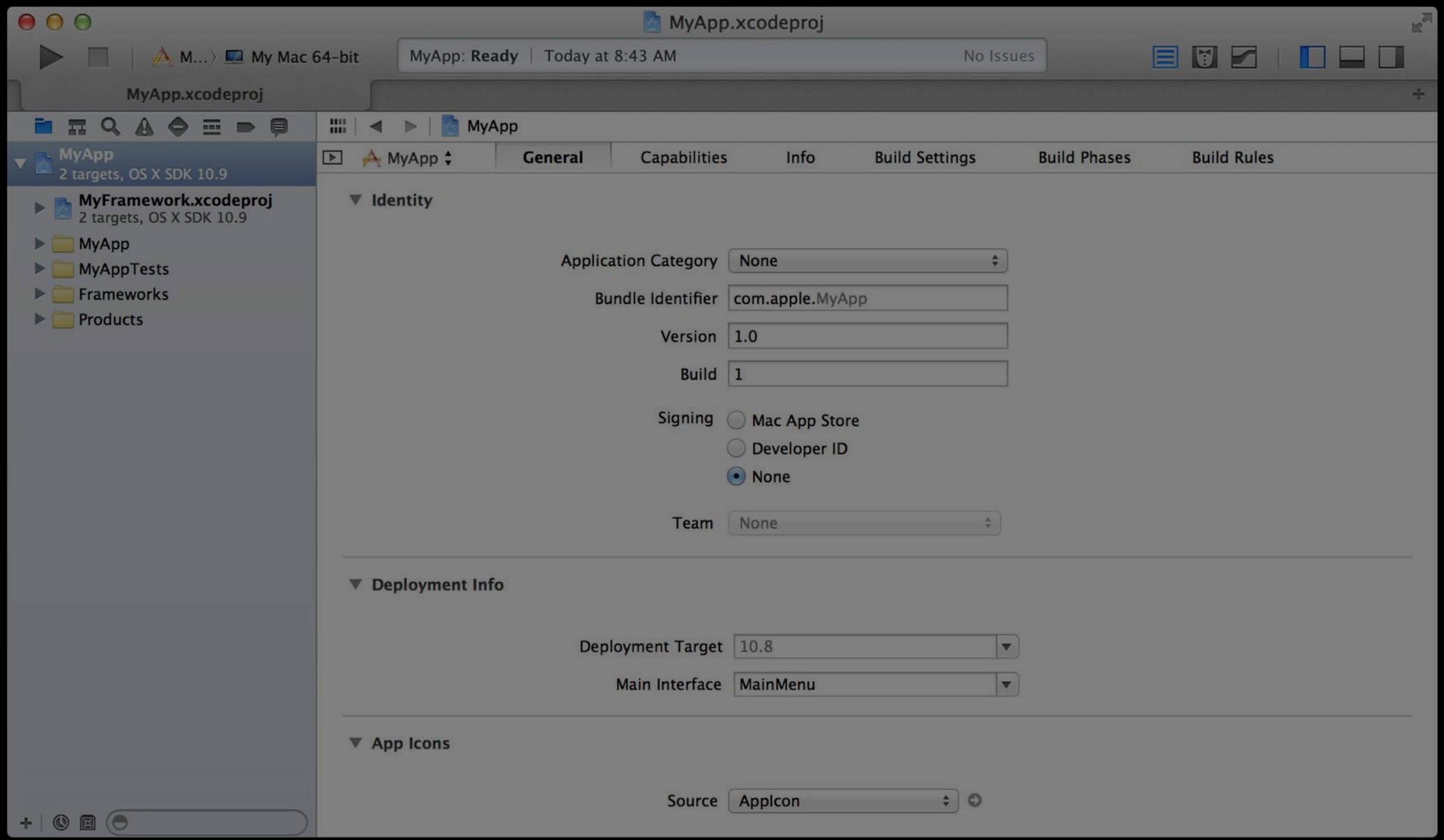
Target Settings



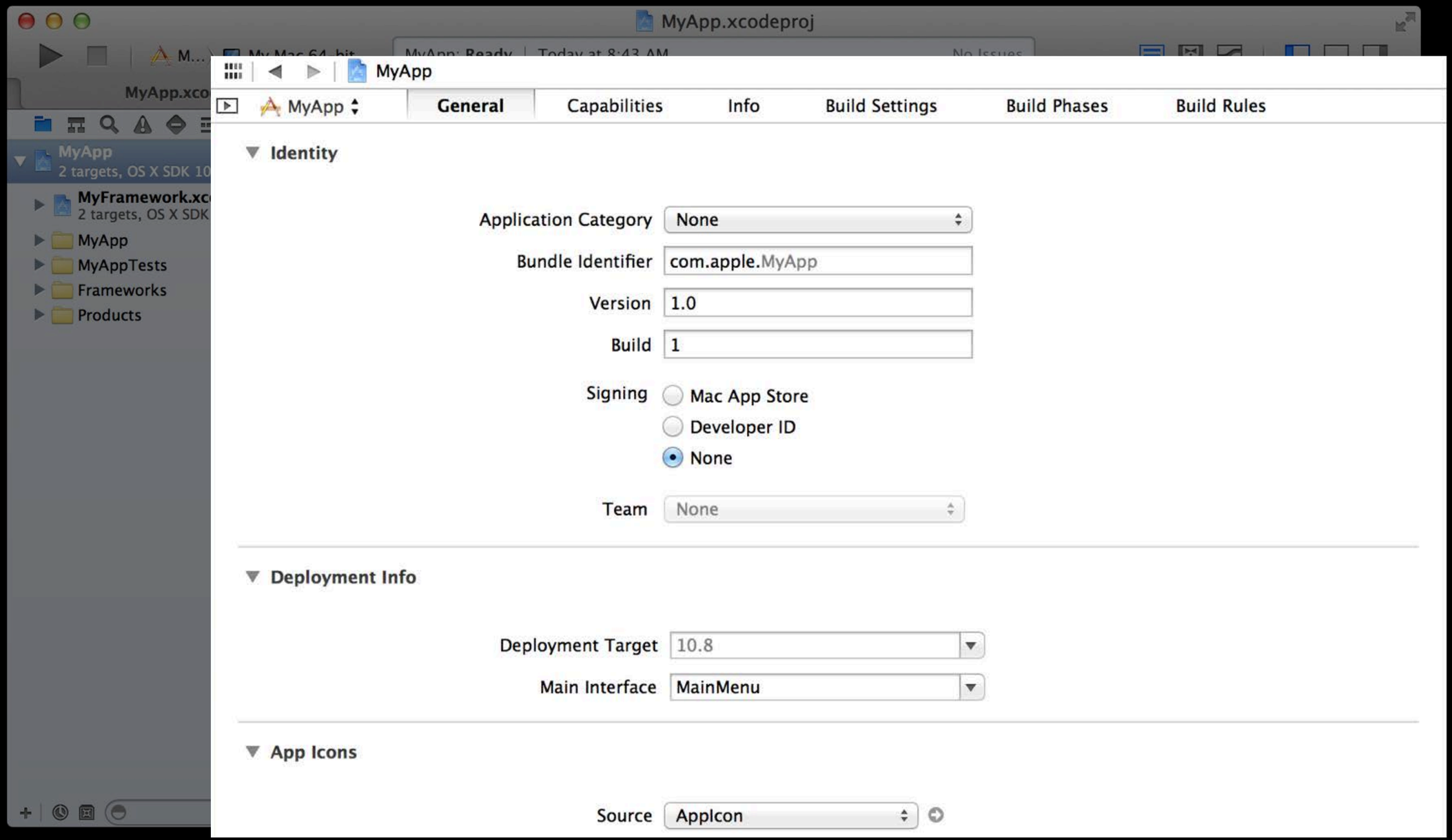
Target Settings



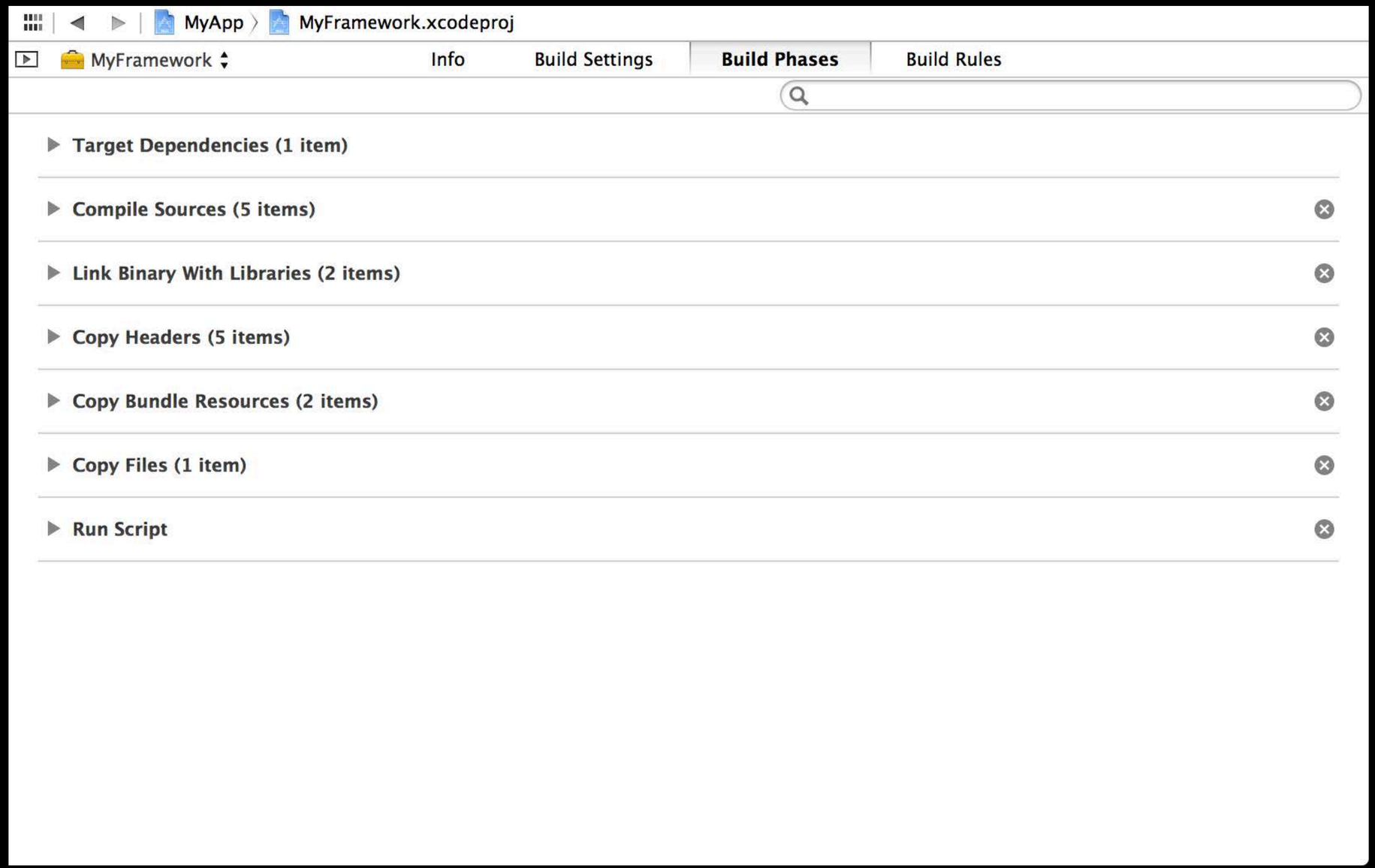
Target Settings



Target Settings

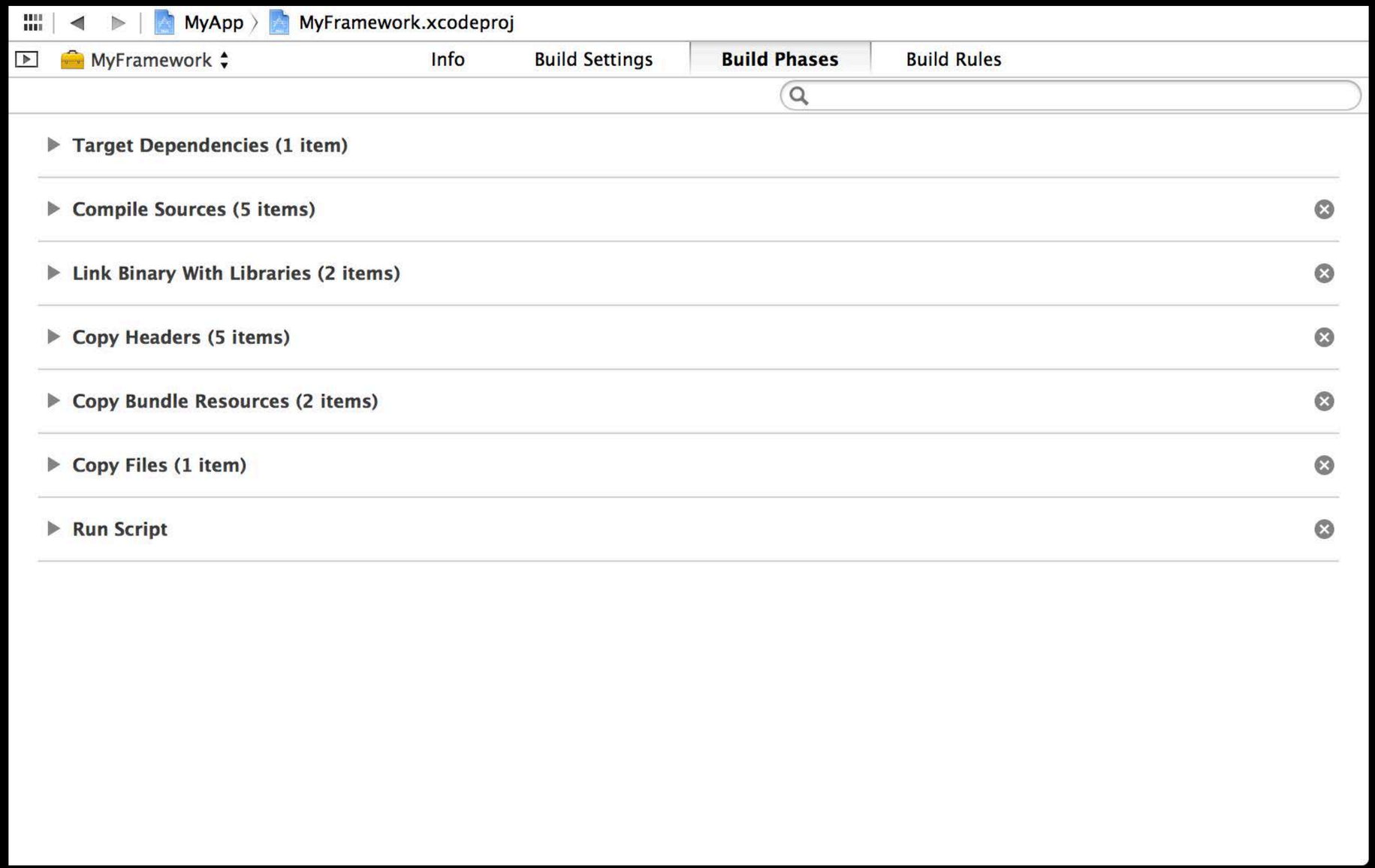


Build Phases



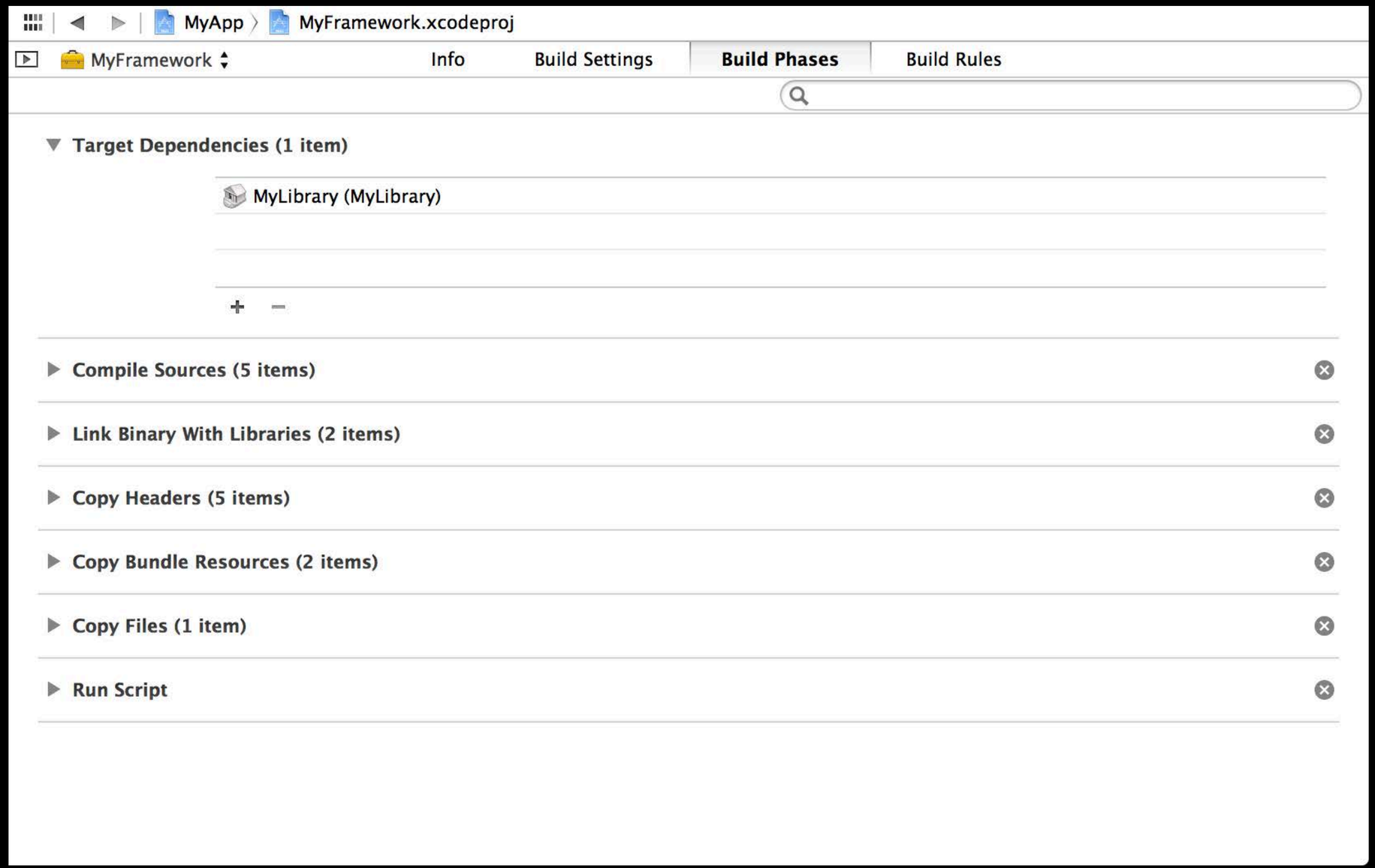
Build Phases

Build Phases



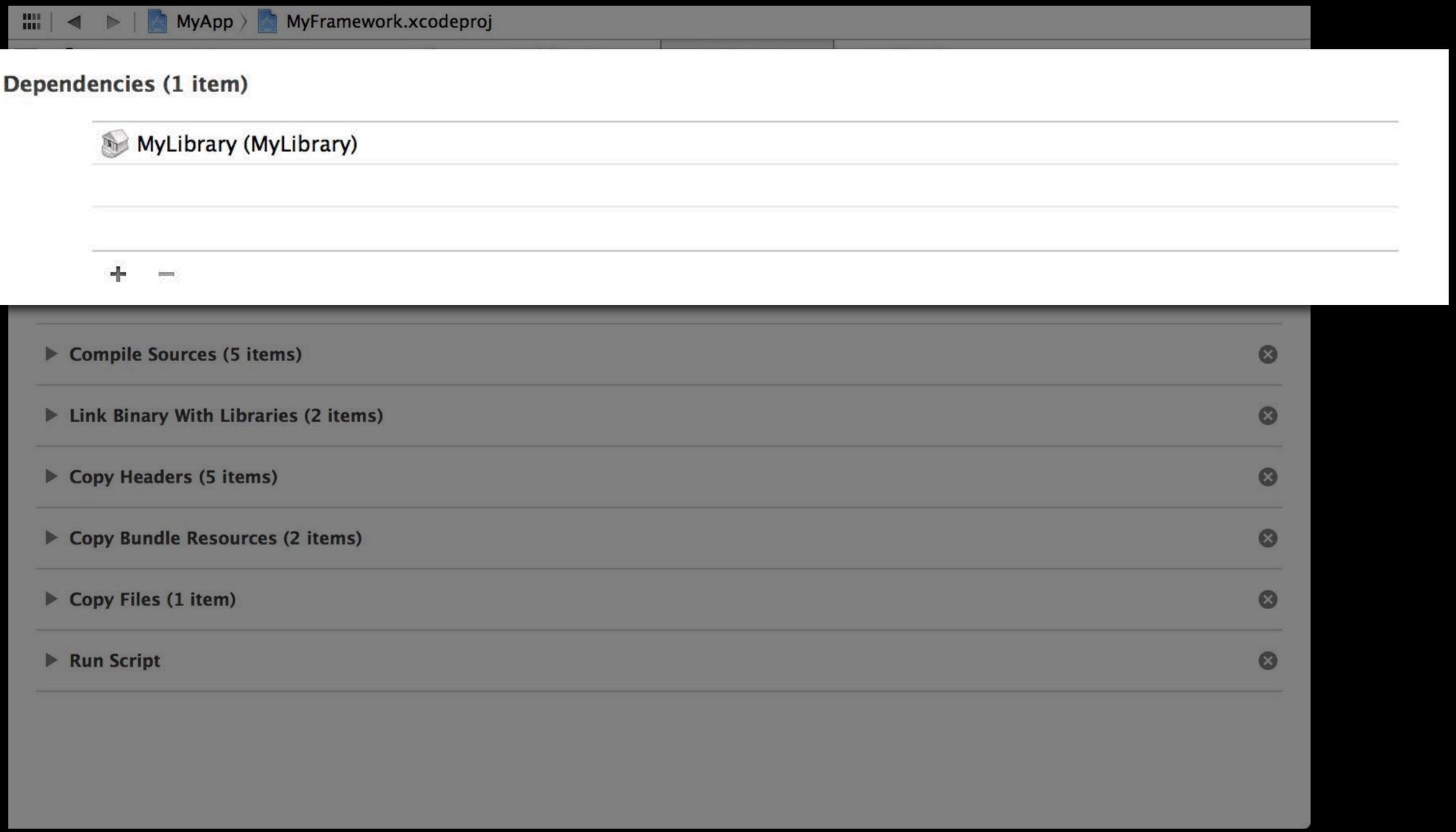
Build Phases

Target Dependencies



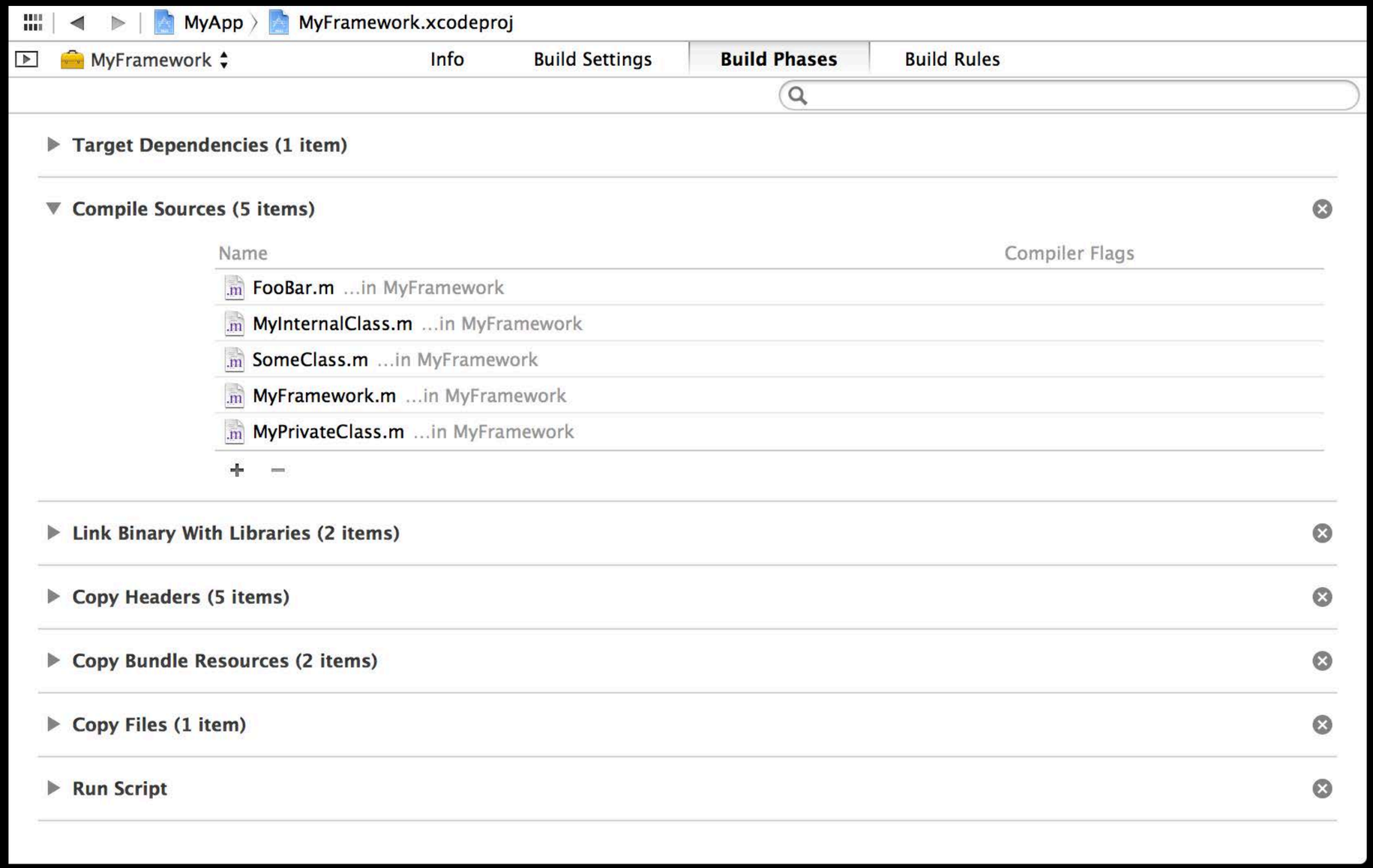
Build Phases

Target Dependencies



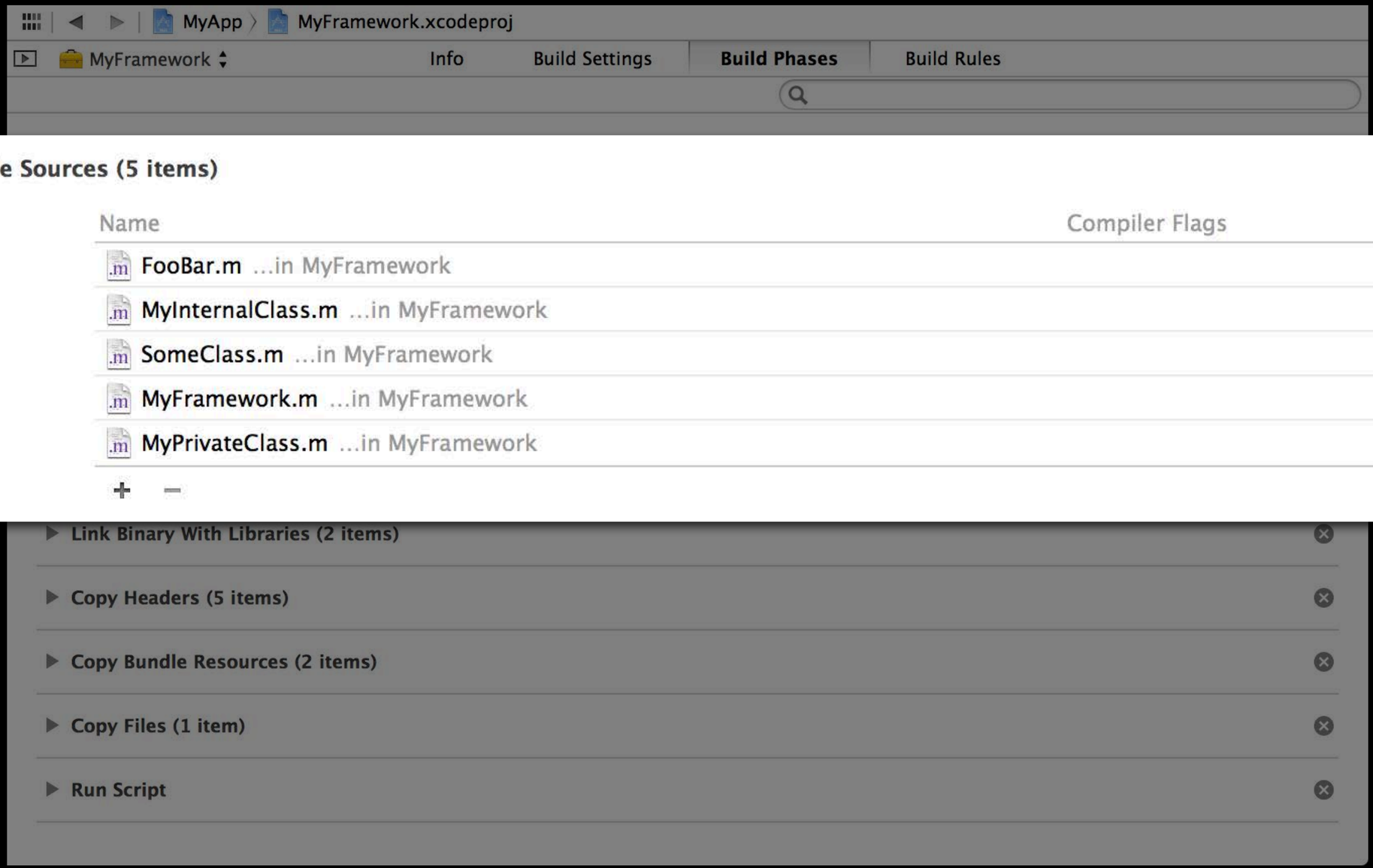
Build Phases

Compile Sources



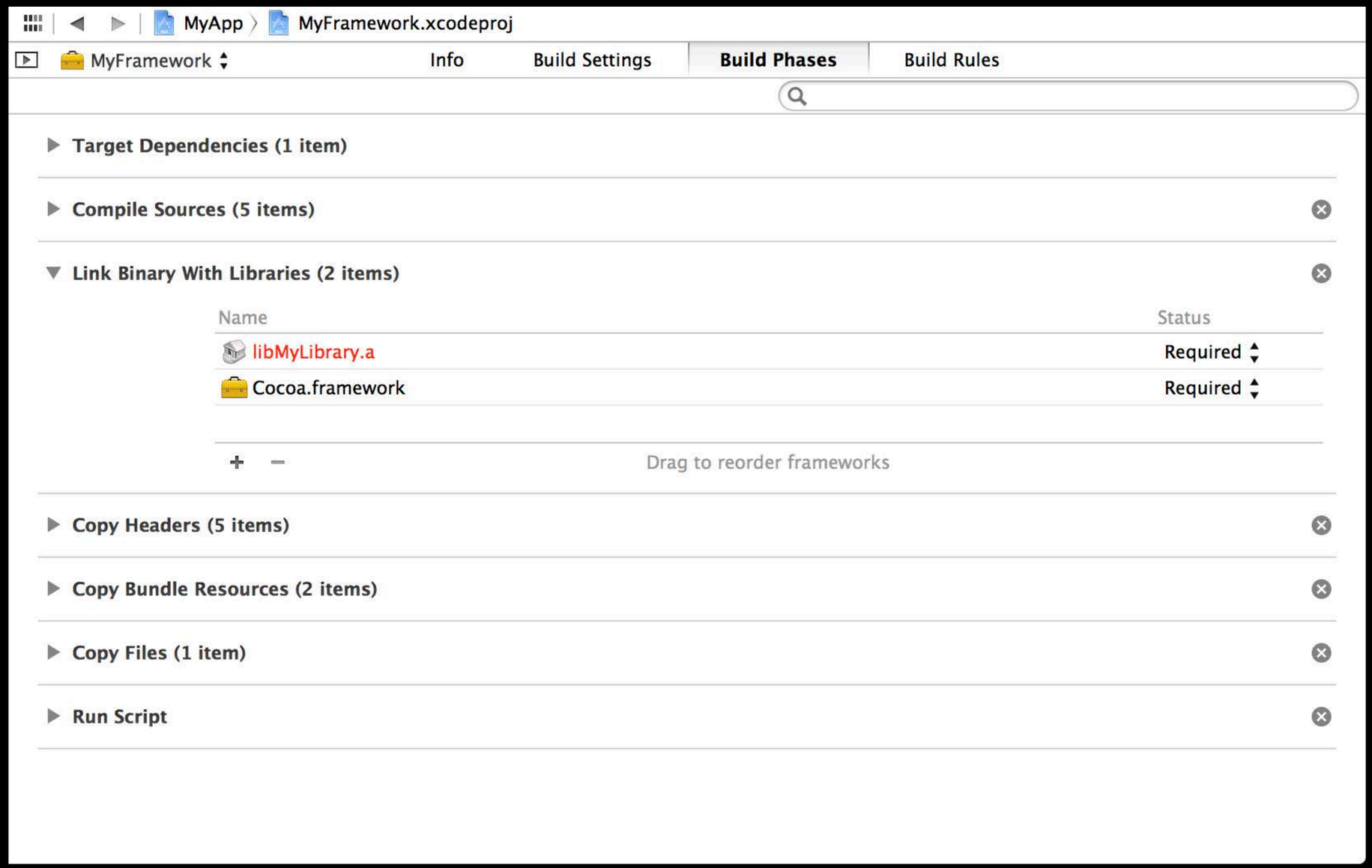
Build Phases

Compile Sources



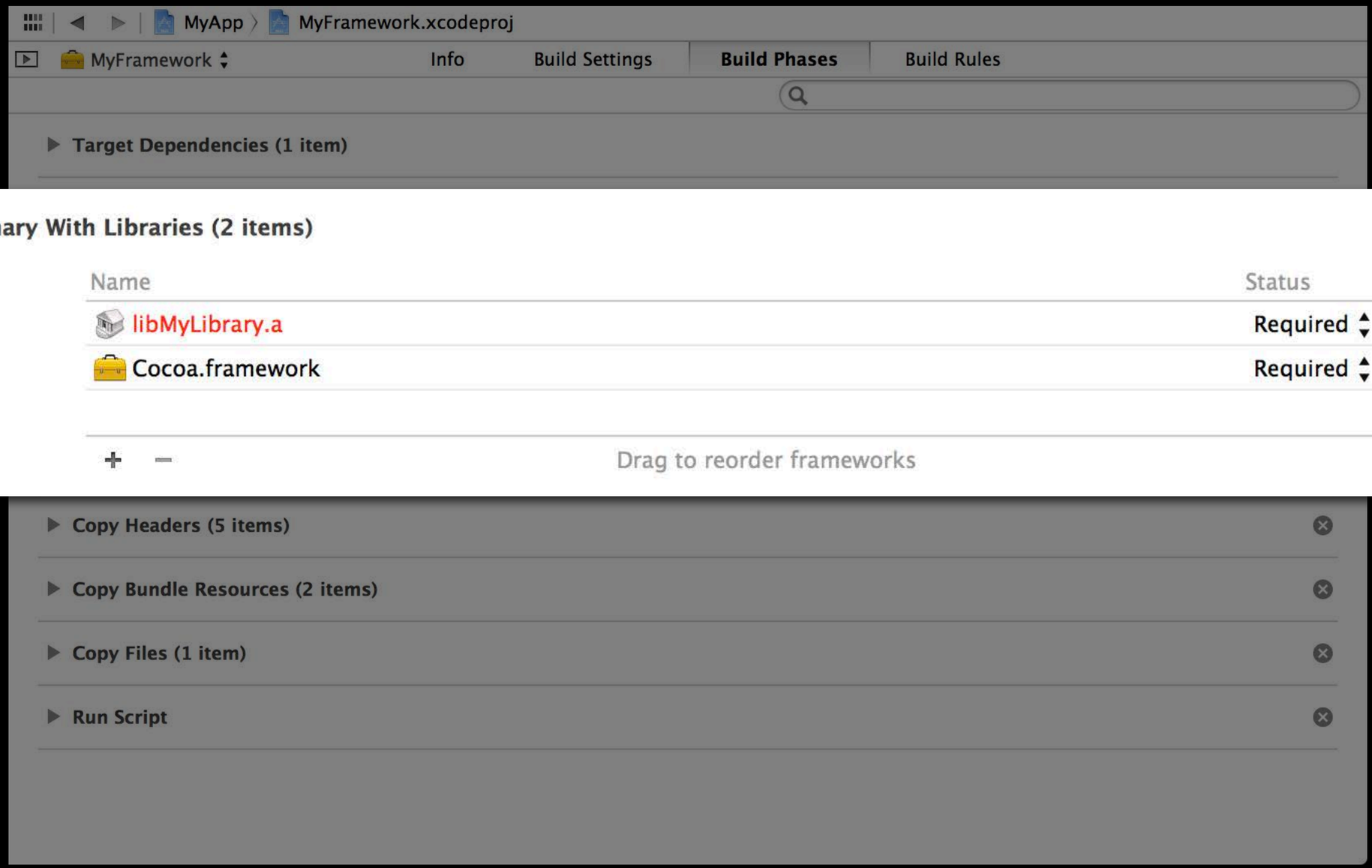
Build Phases

Link Binary with Libraries



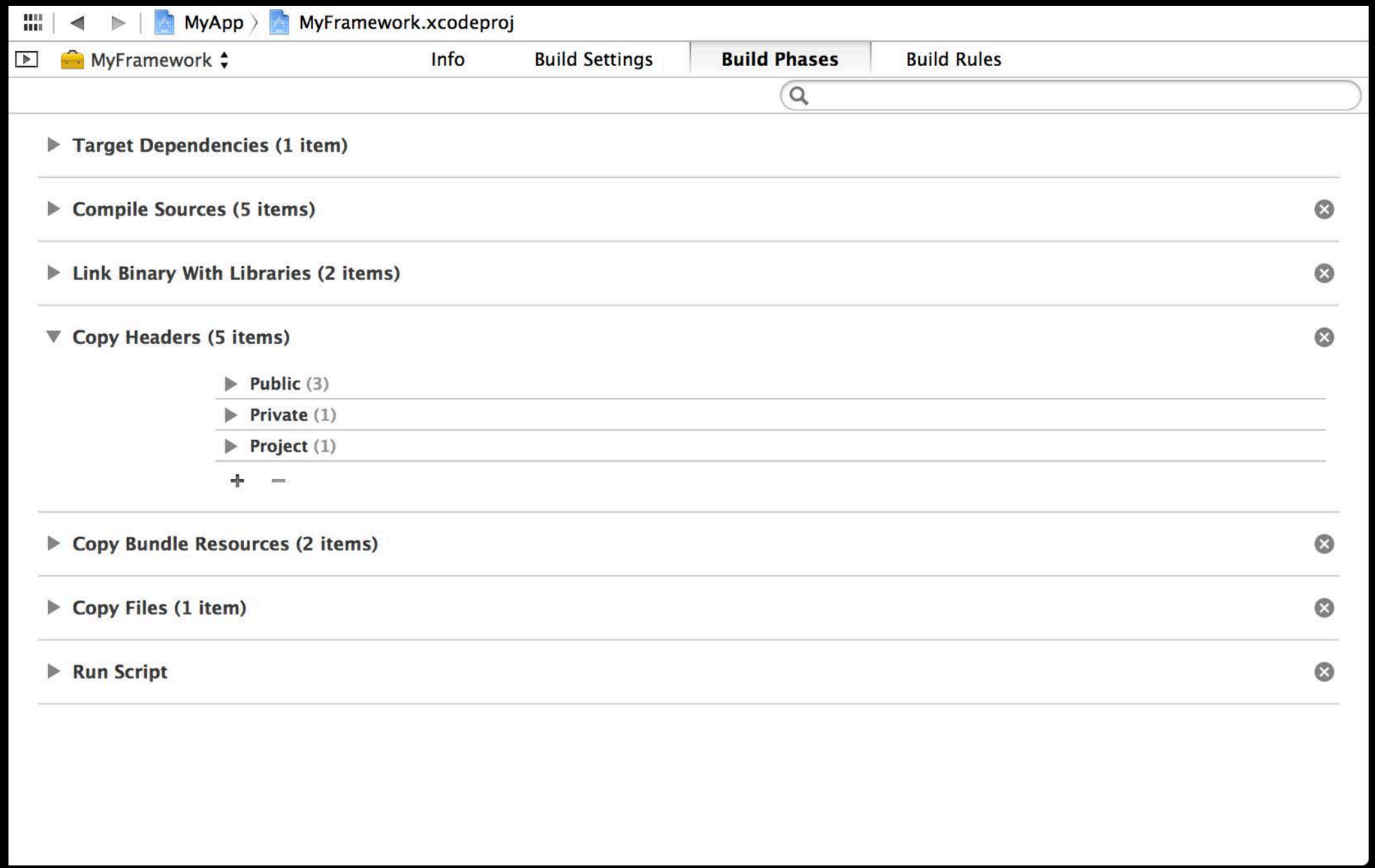
Build Phases

Link Binary with Libraries



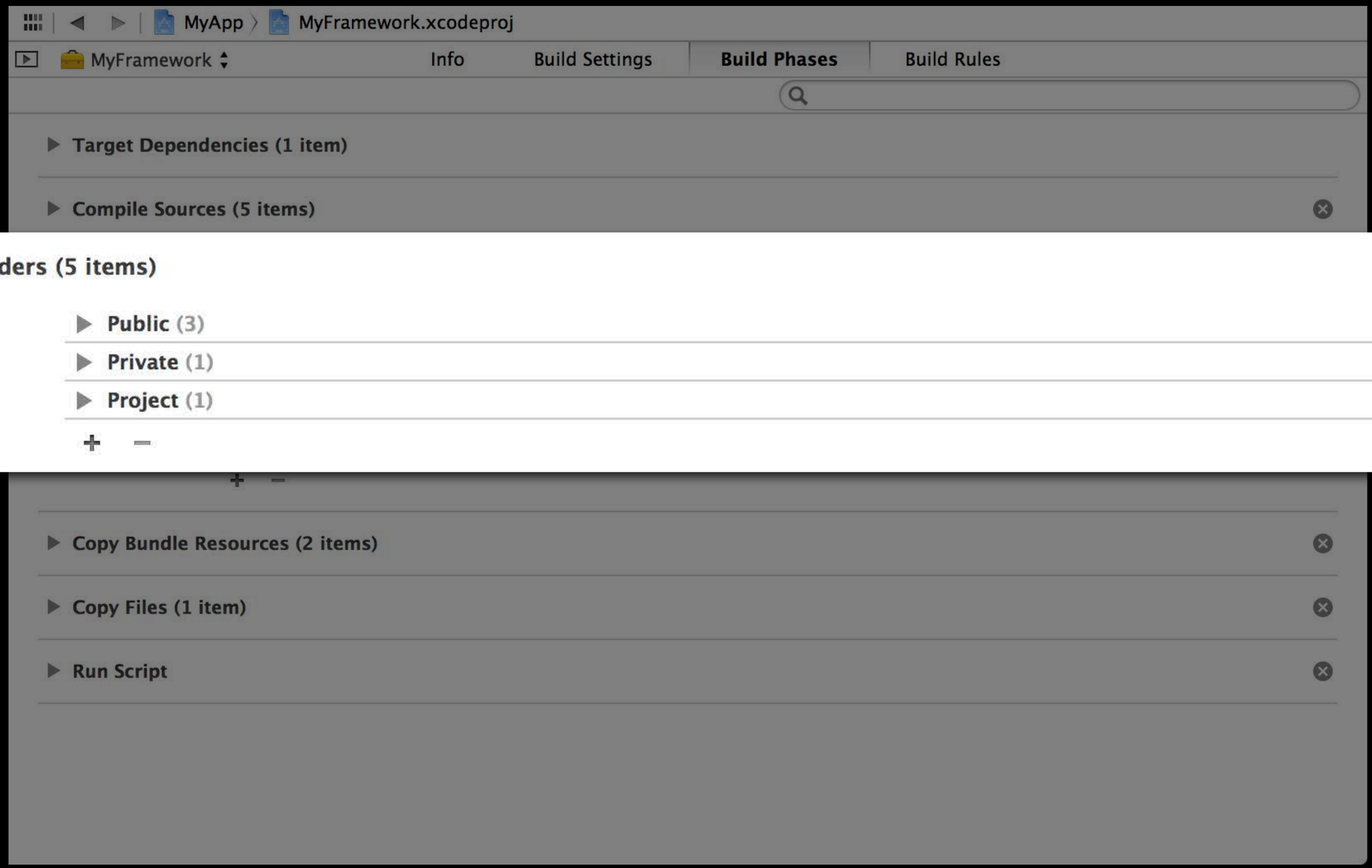
Build Phases

Copy Headers



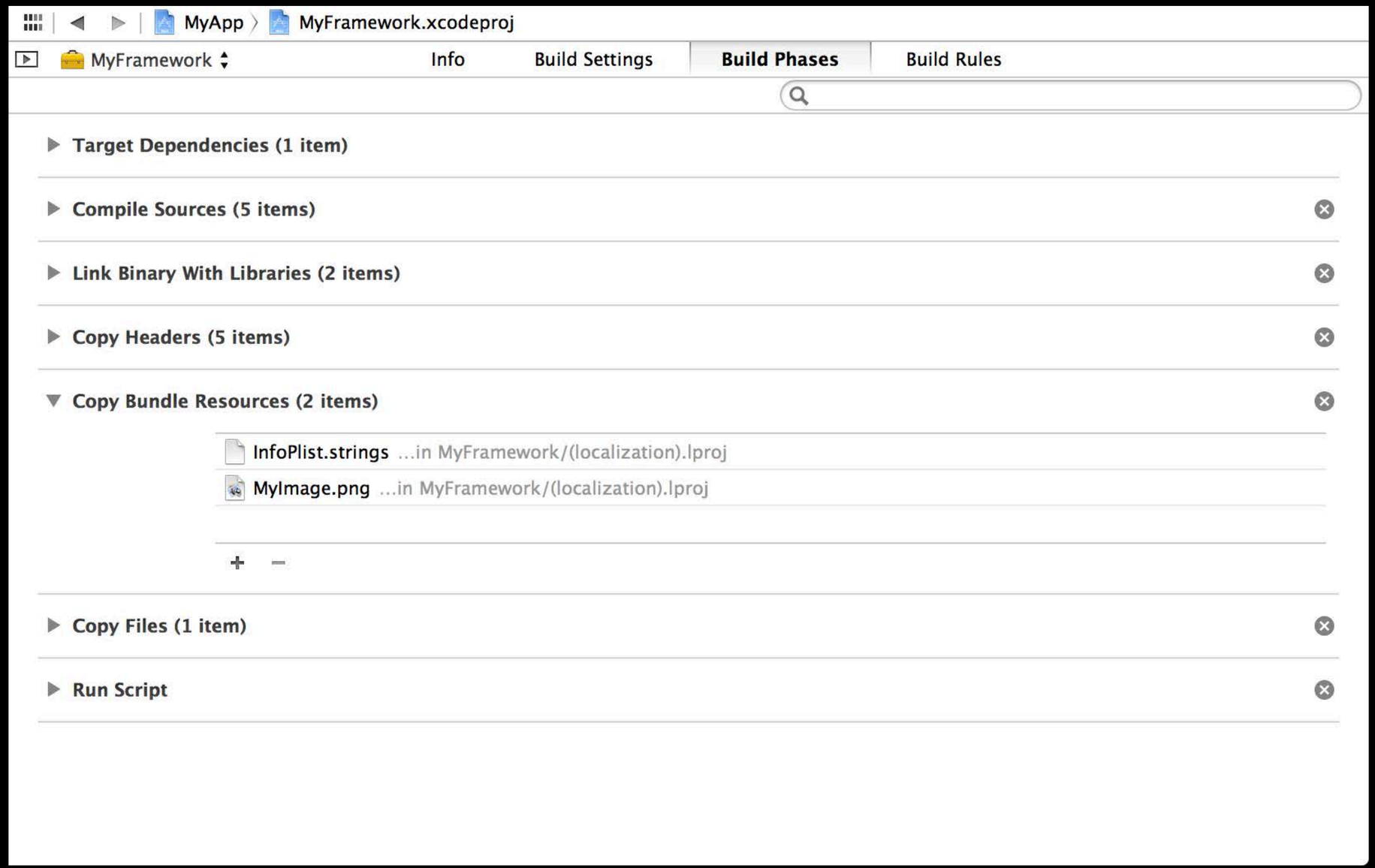
Build Phases

Copy Headers



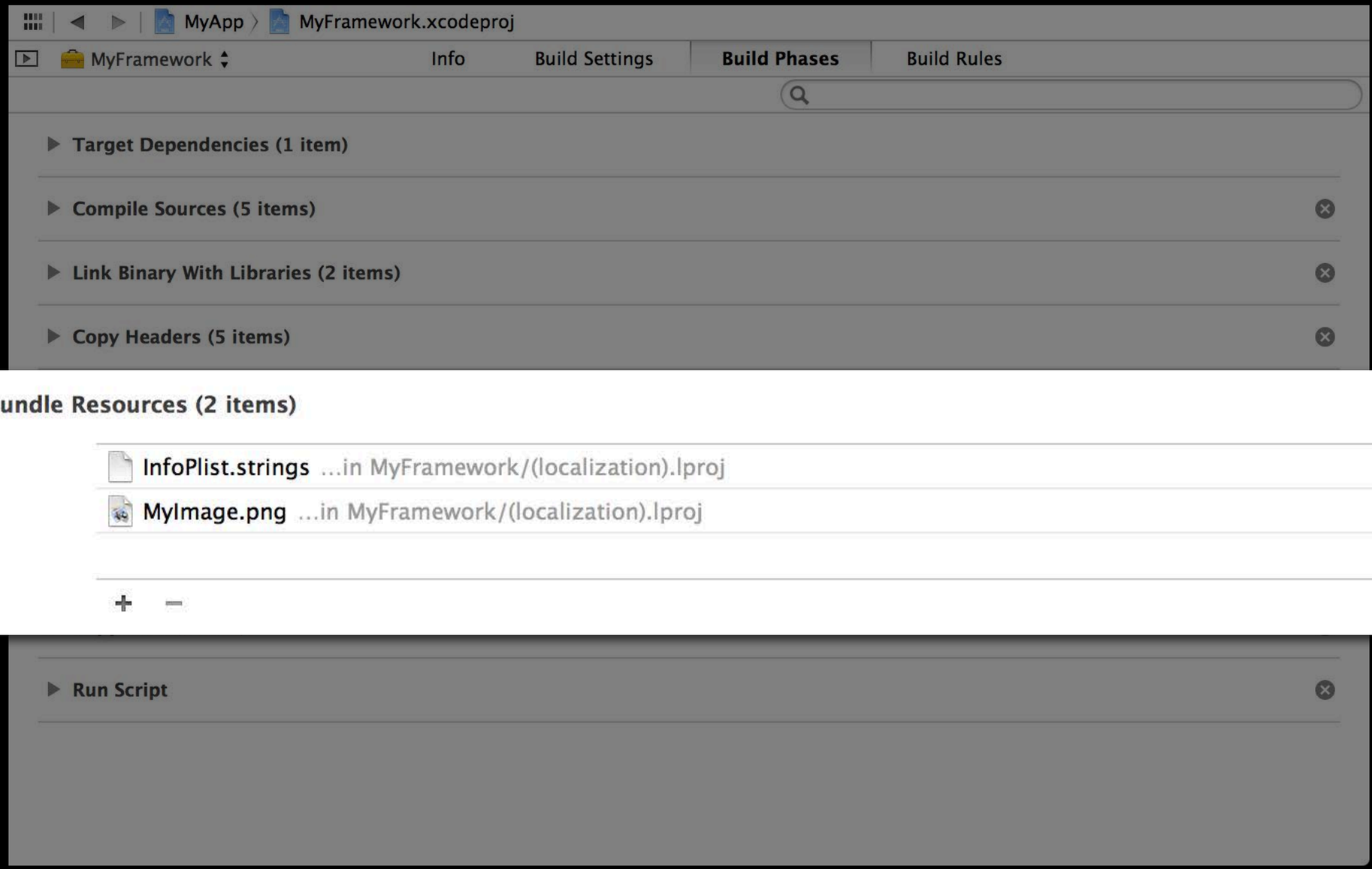
Build Phases

Copy Bundle Resources



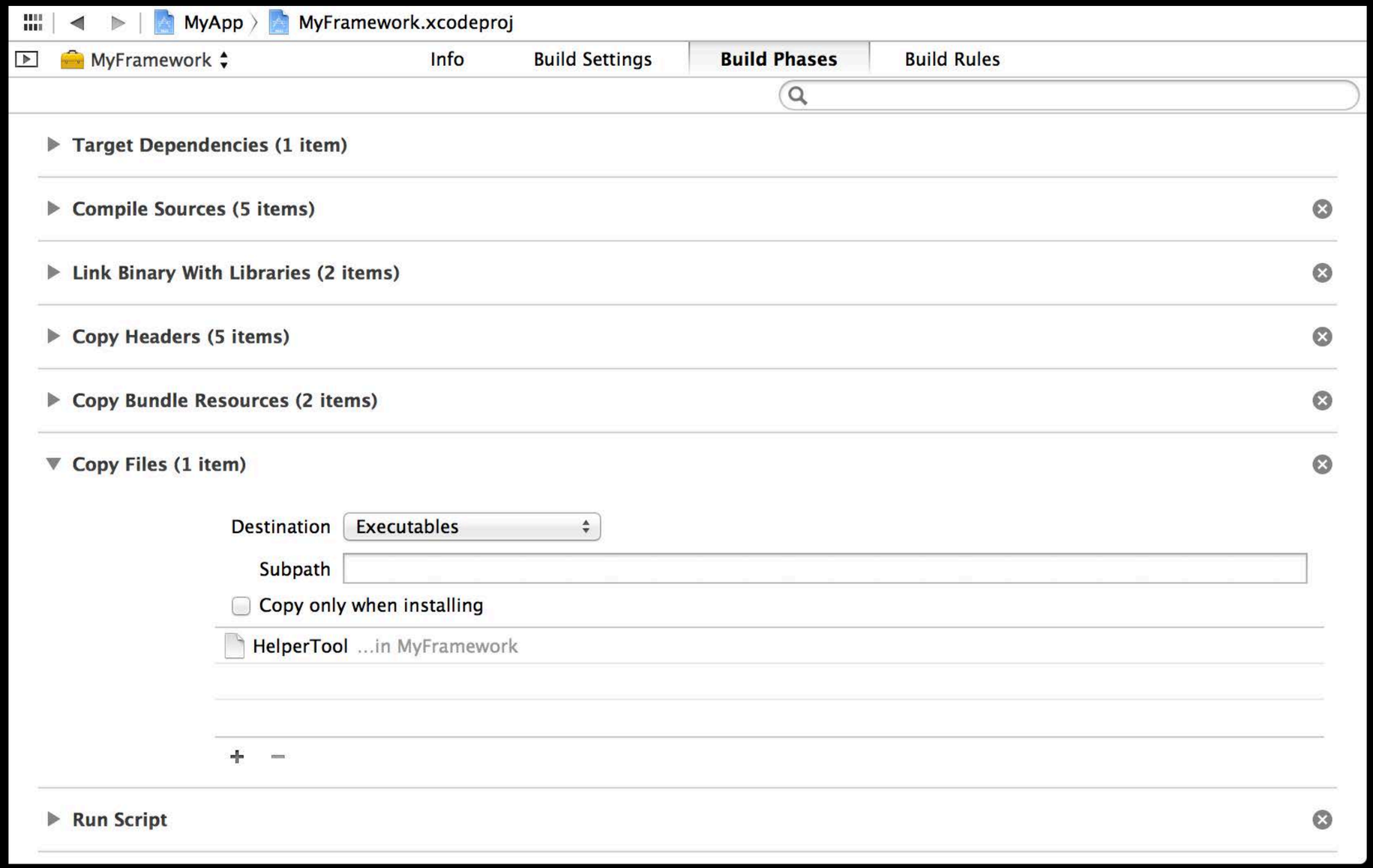
Build Phases

Copy Bundle Resources



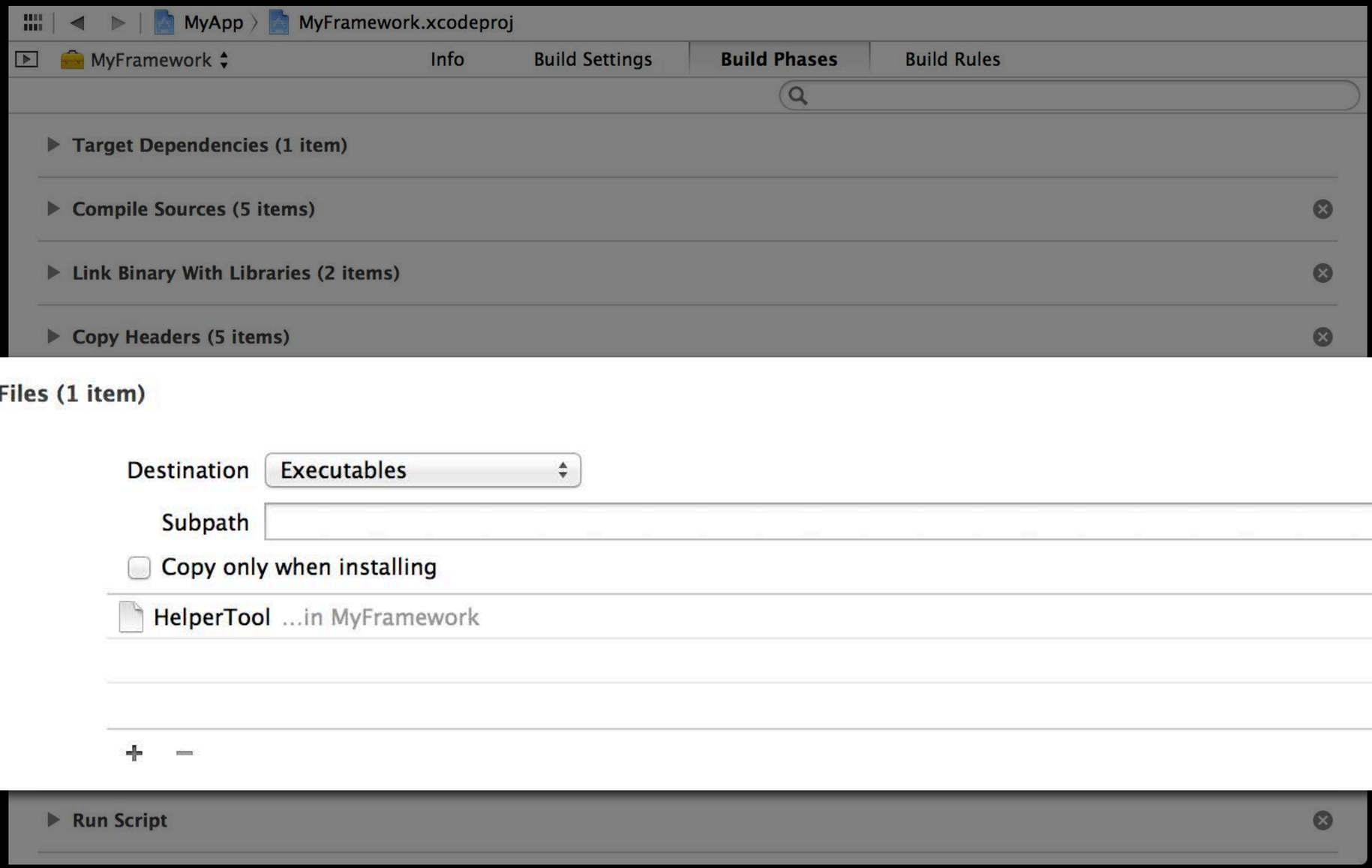
Build Phases

Copy Files



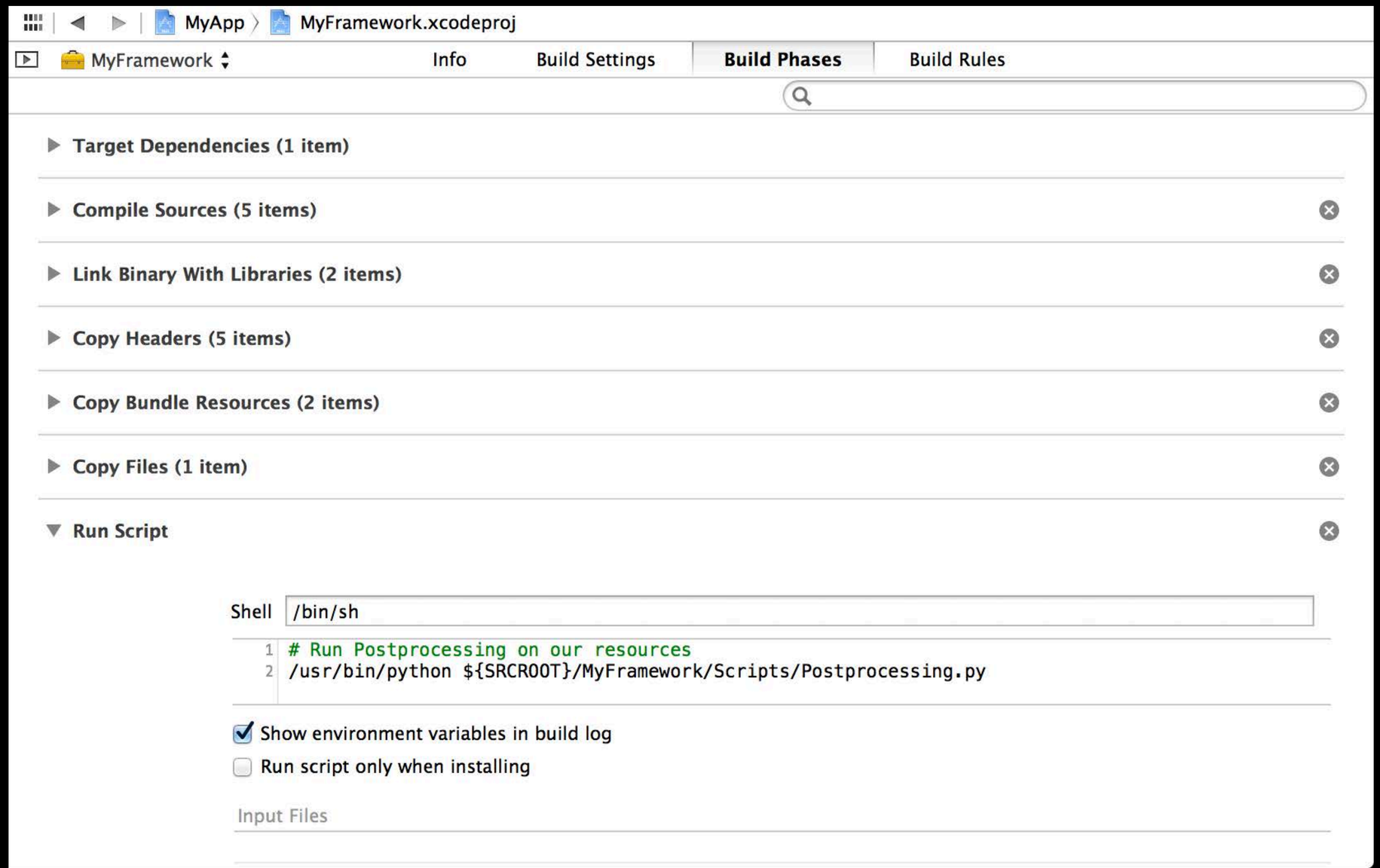
Build Phases

Copy Files



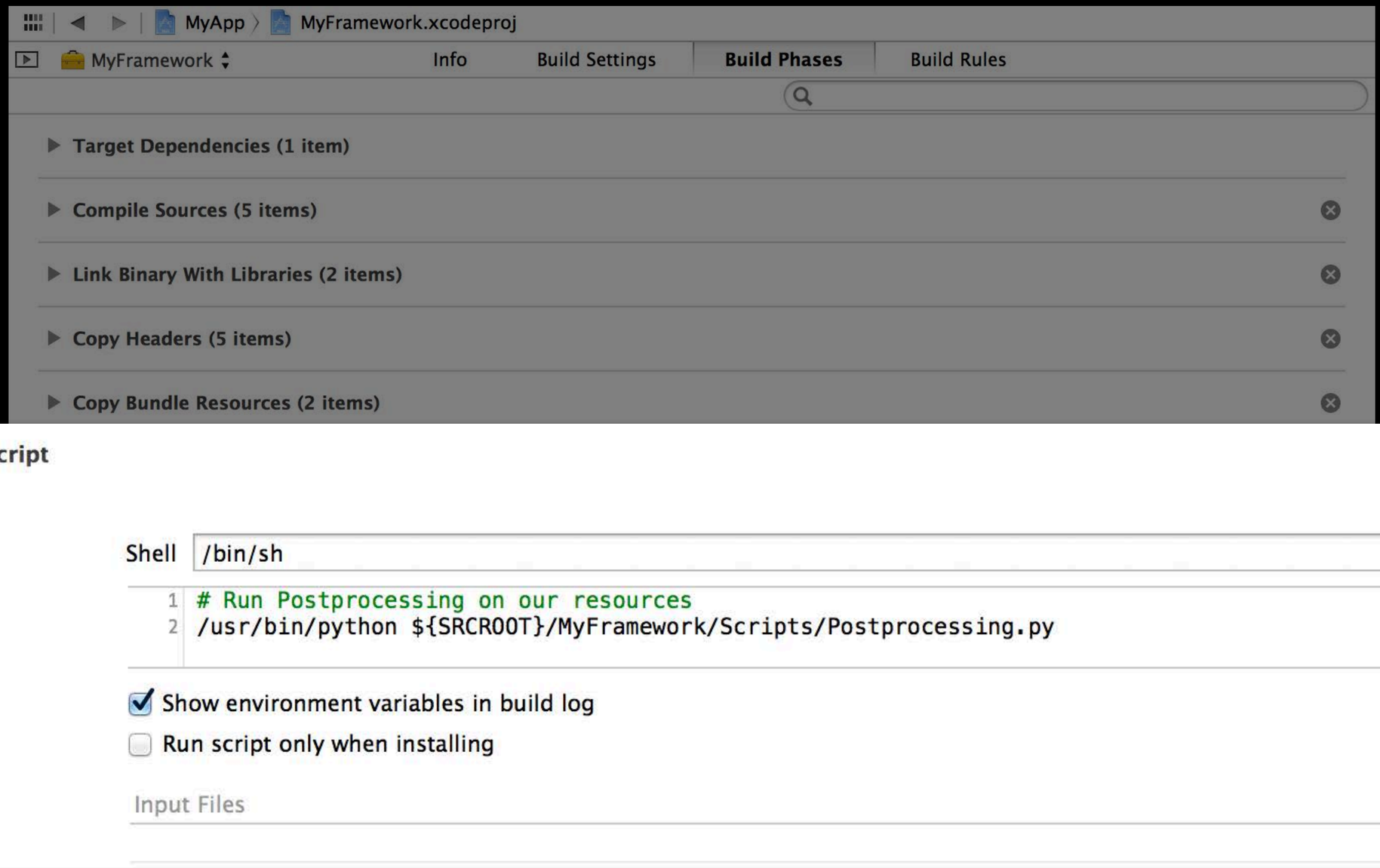
Build Phases

Run Script



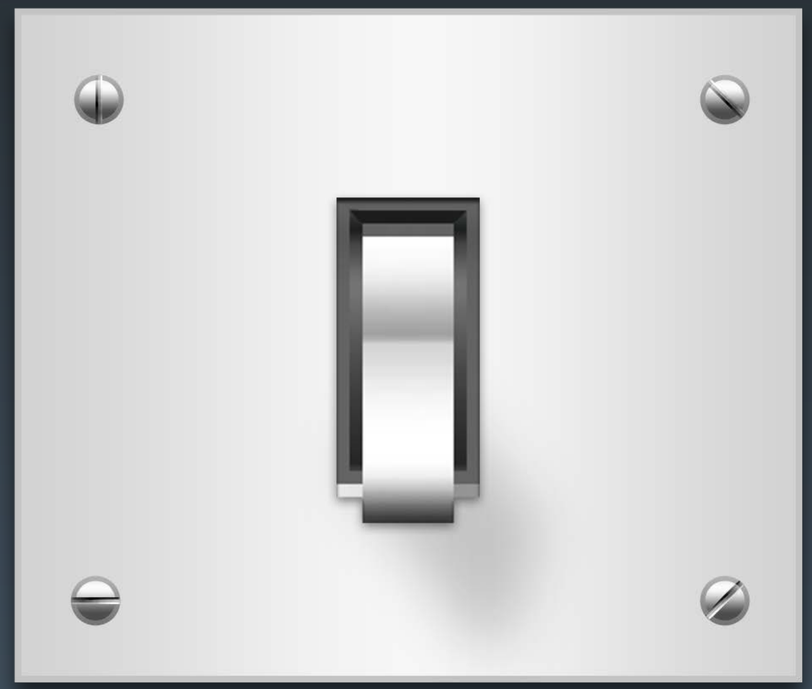
Build Phases

Run Script



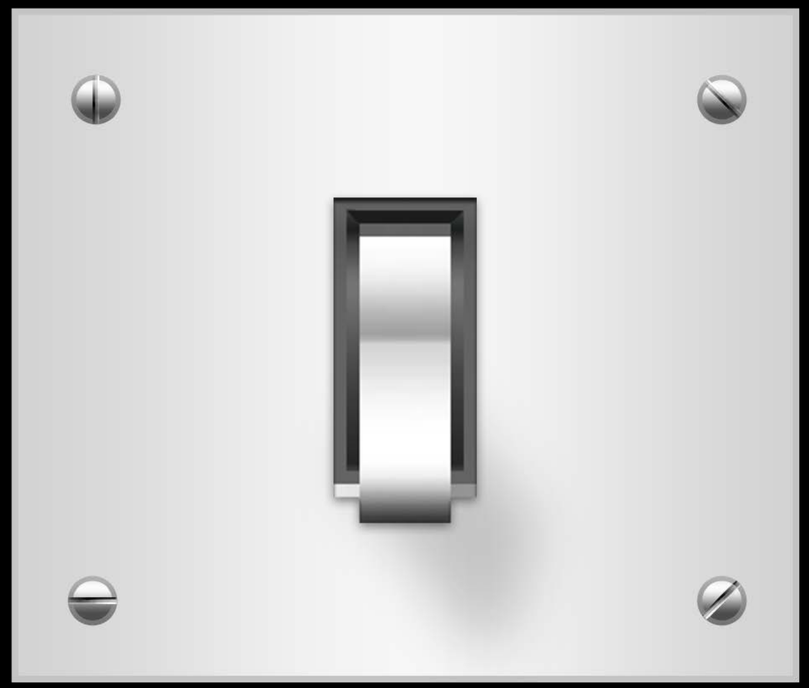
Build Settings

Values that control how a product is built

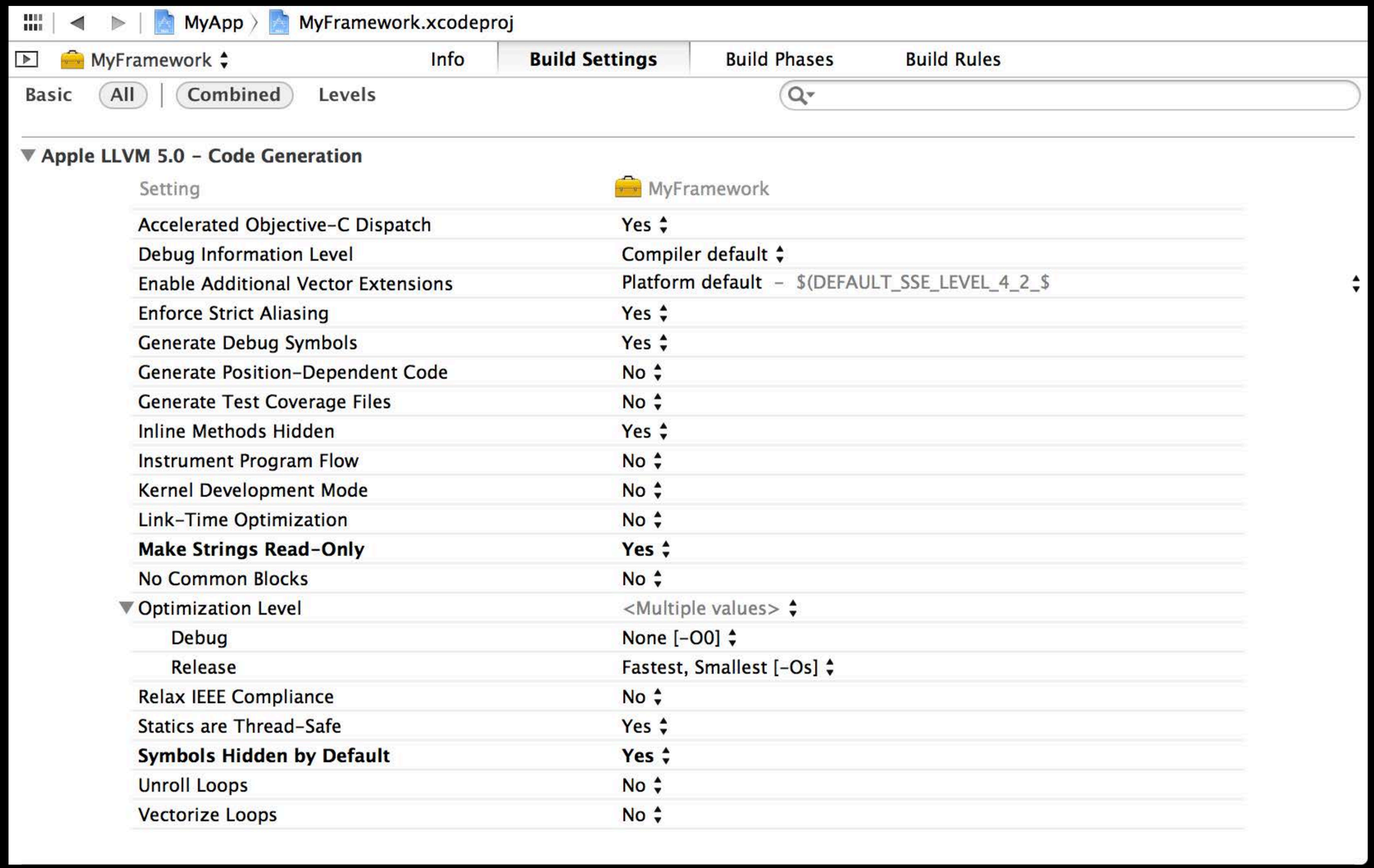


Why Use Build Settings?

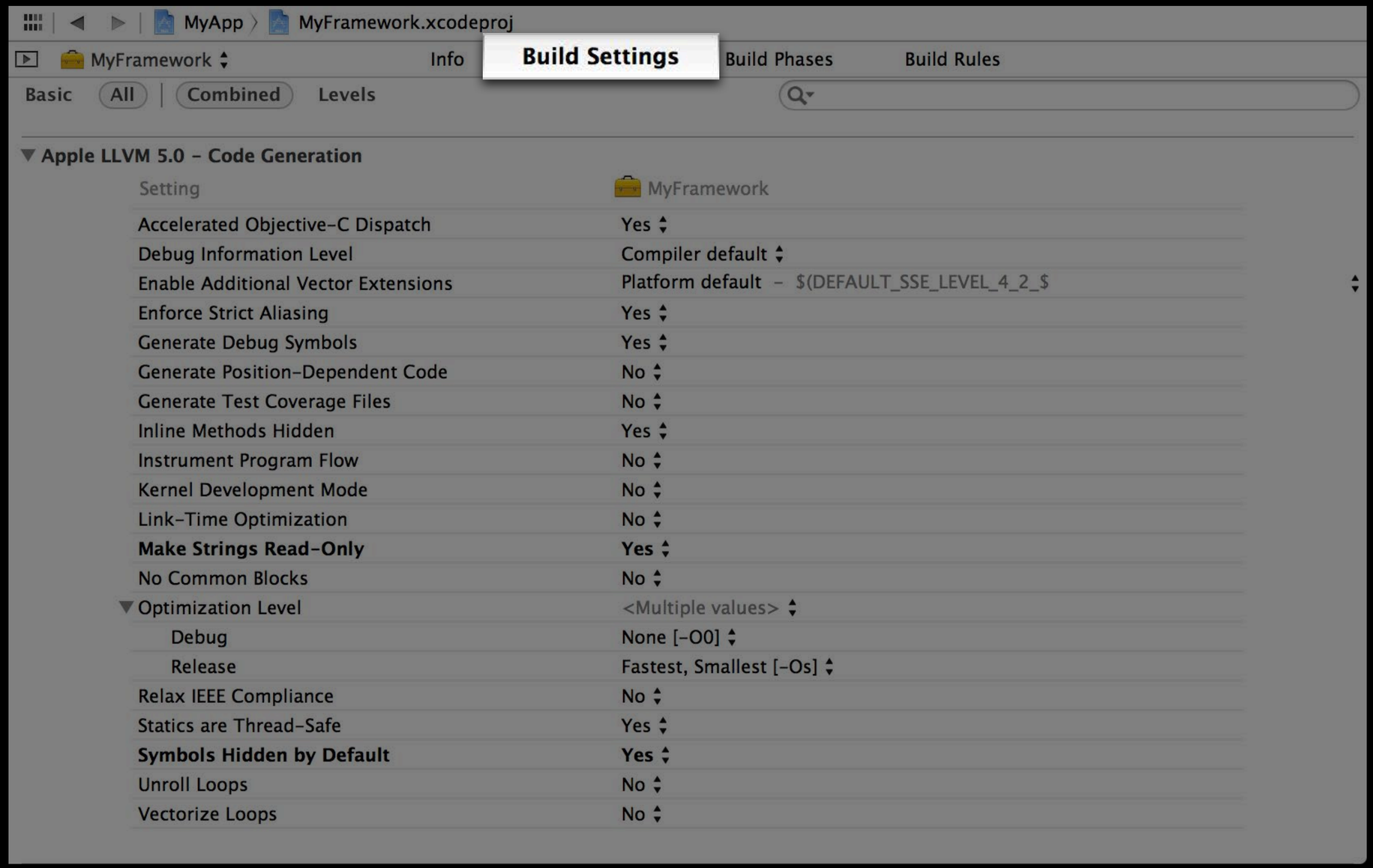
- To change how a product is built
  - Product name
  - SDK
  - Preprocessor defines
  - Warning flags
  - Many more



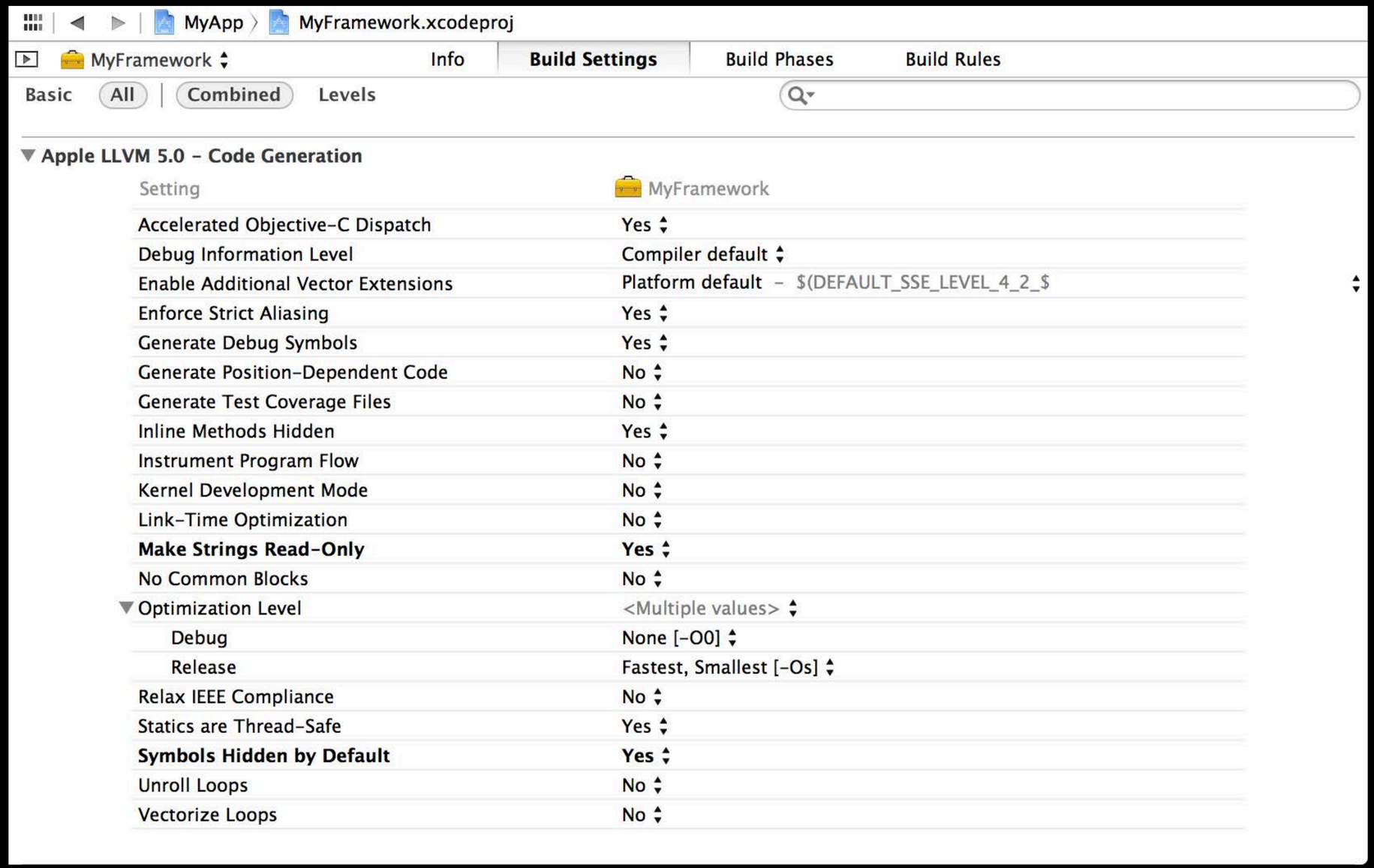
Build Settings



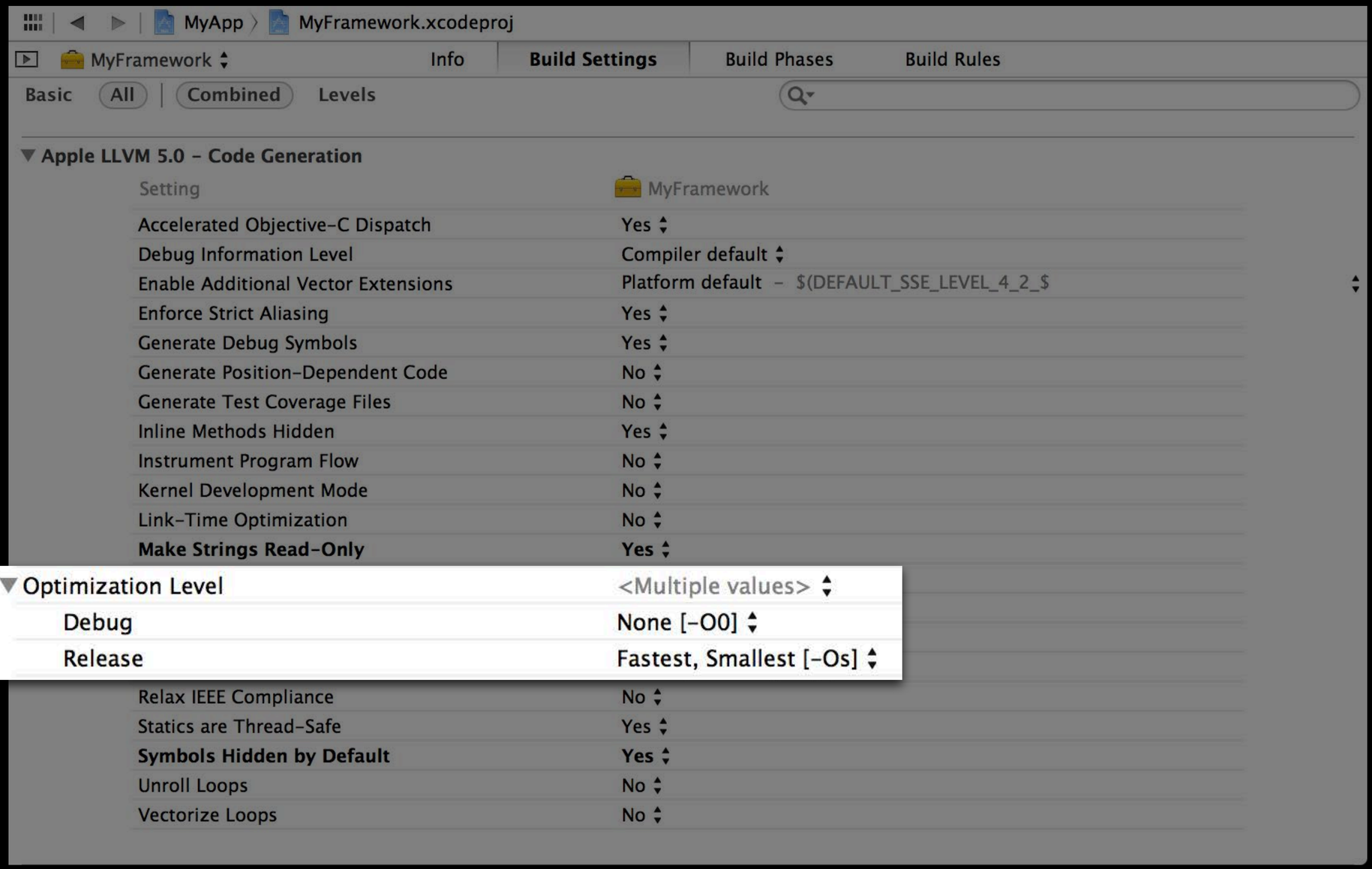
Build Settings



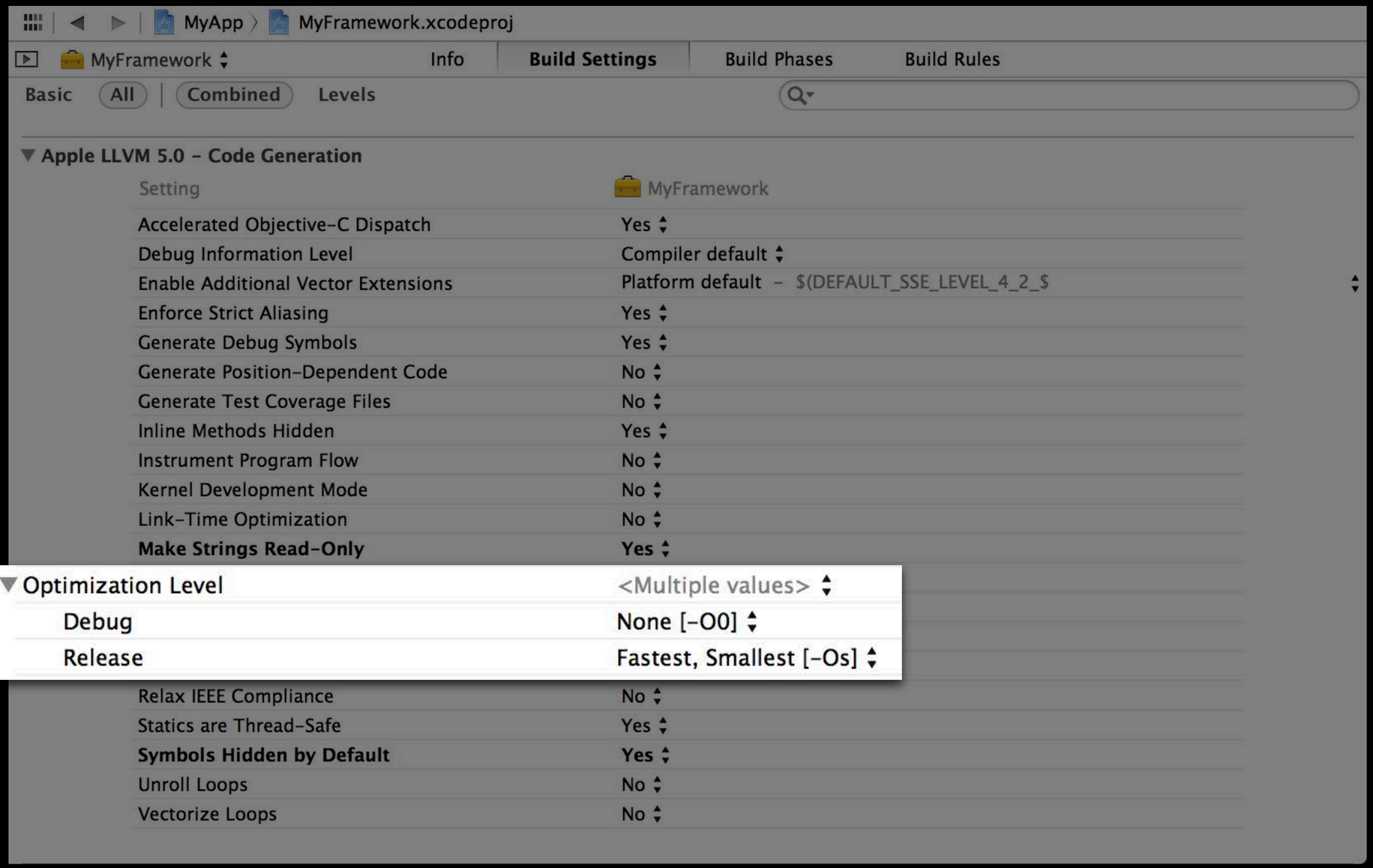
Build Settings



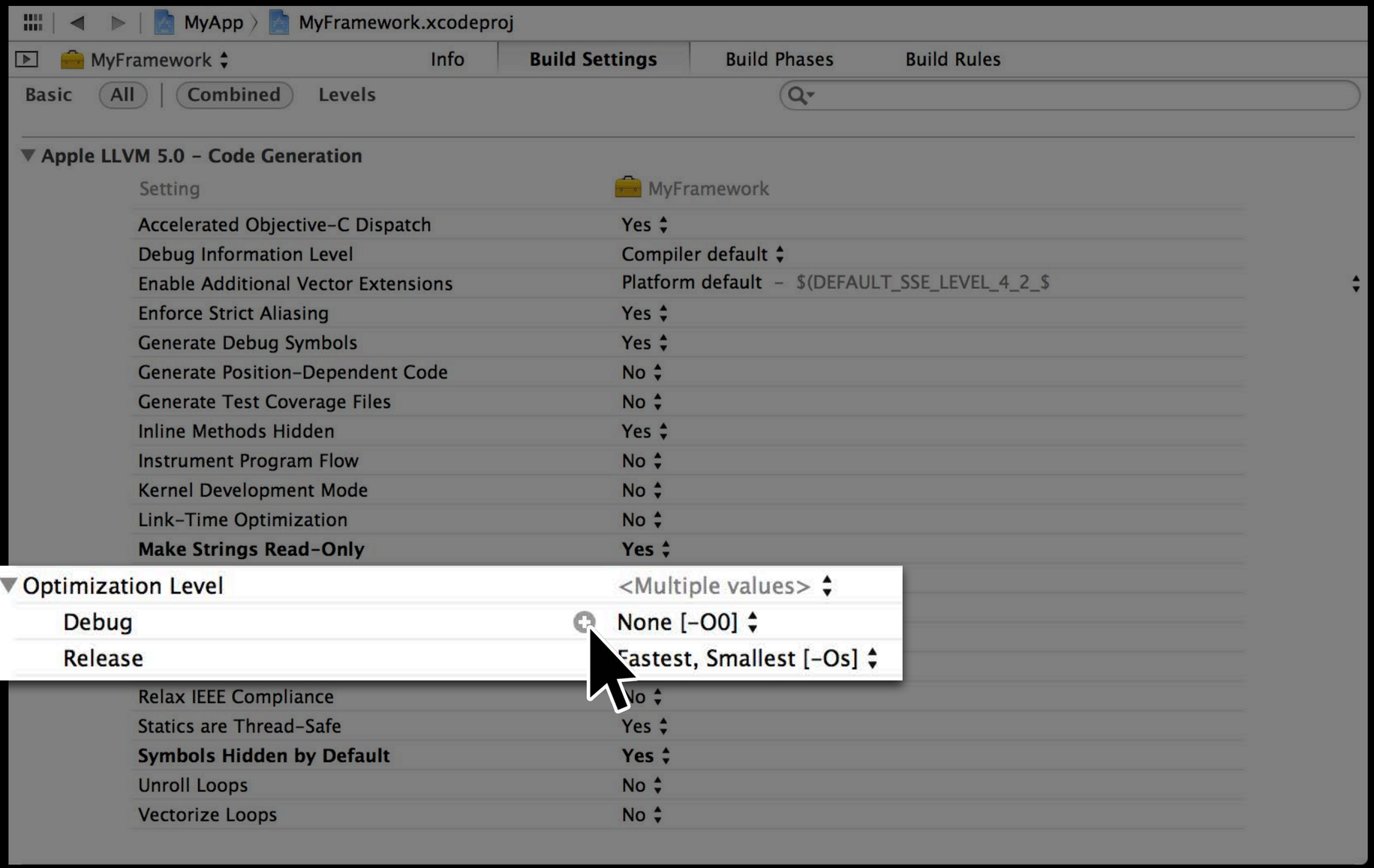
Build Settings



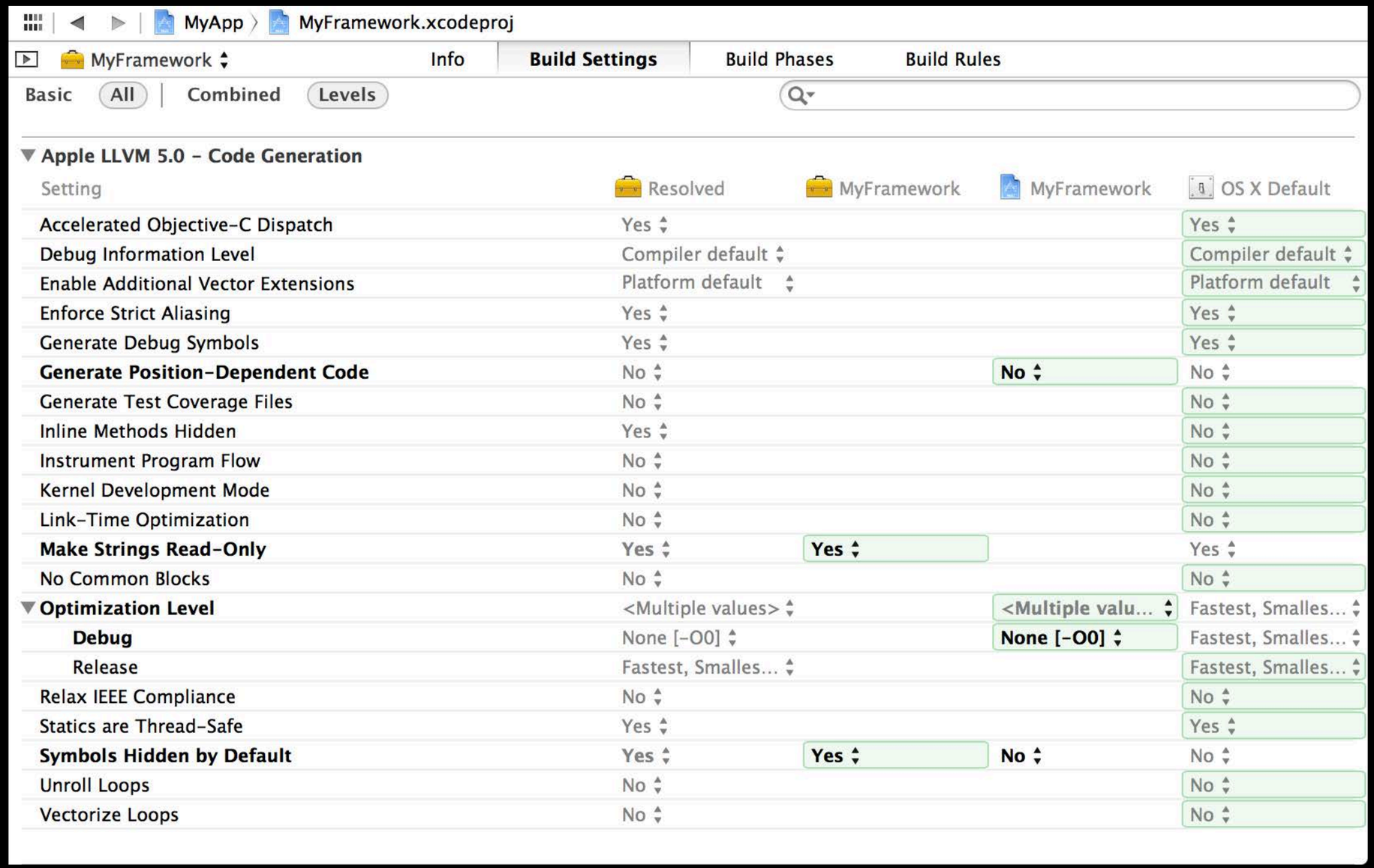
Build Settings



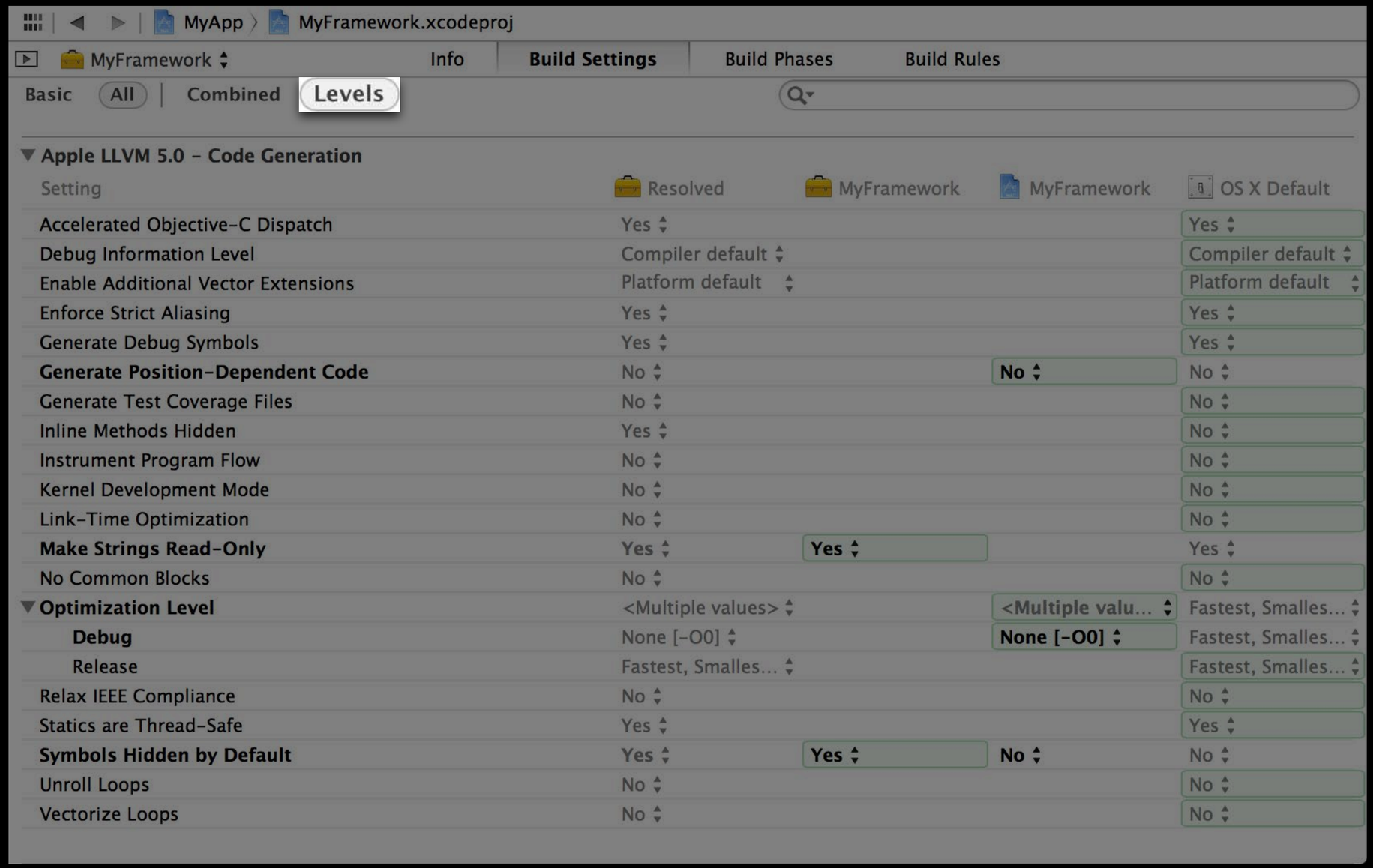
Build Settings



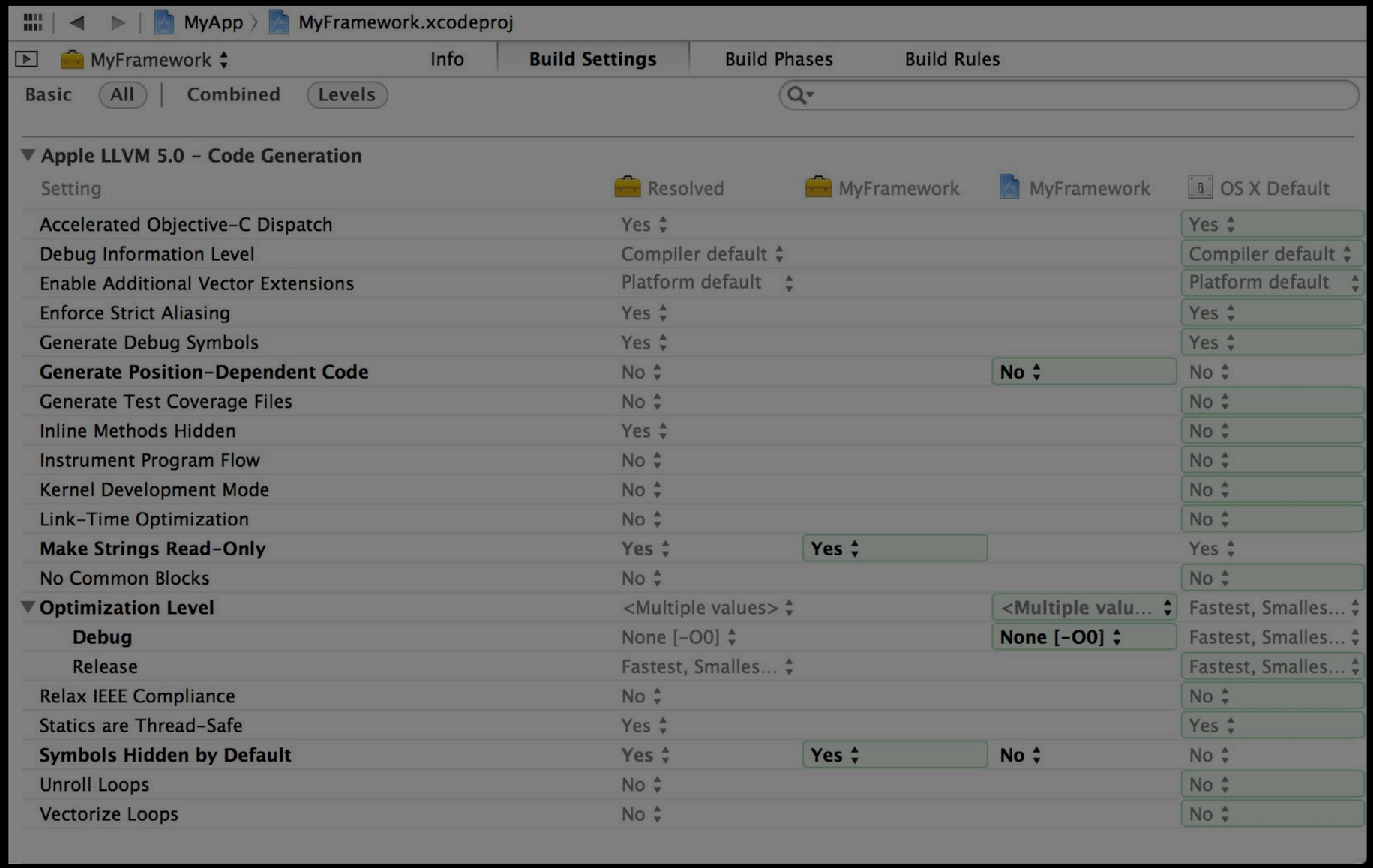
Build Settings



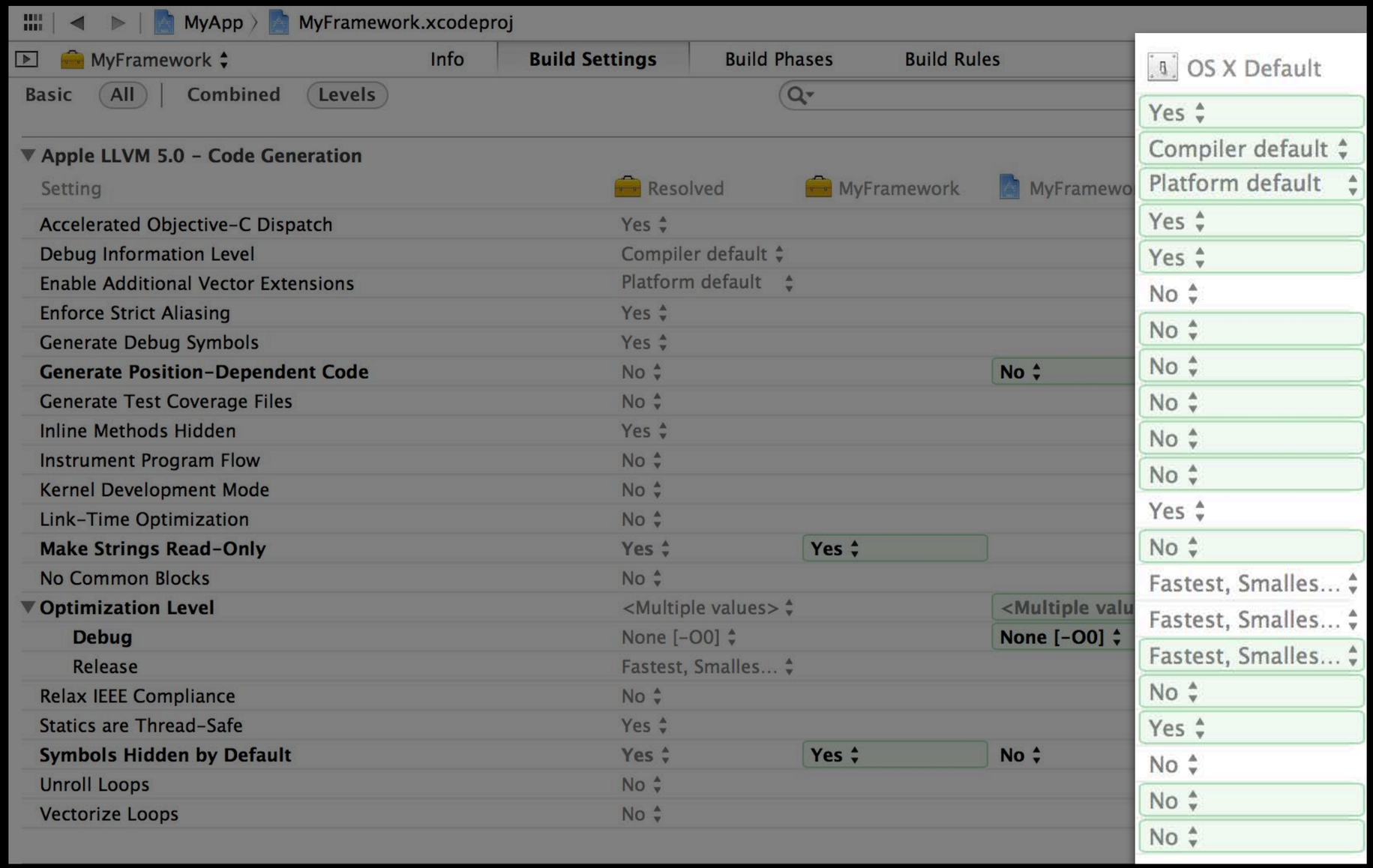
Build Settings



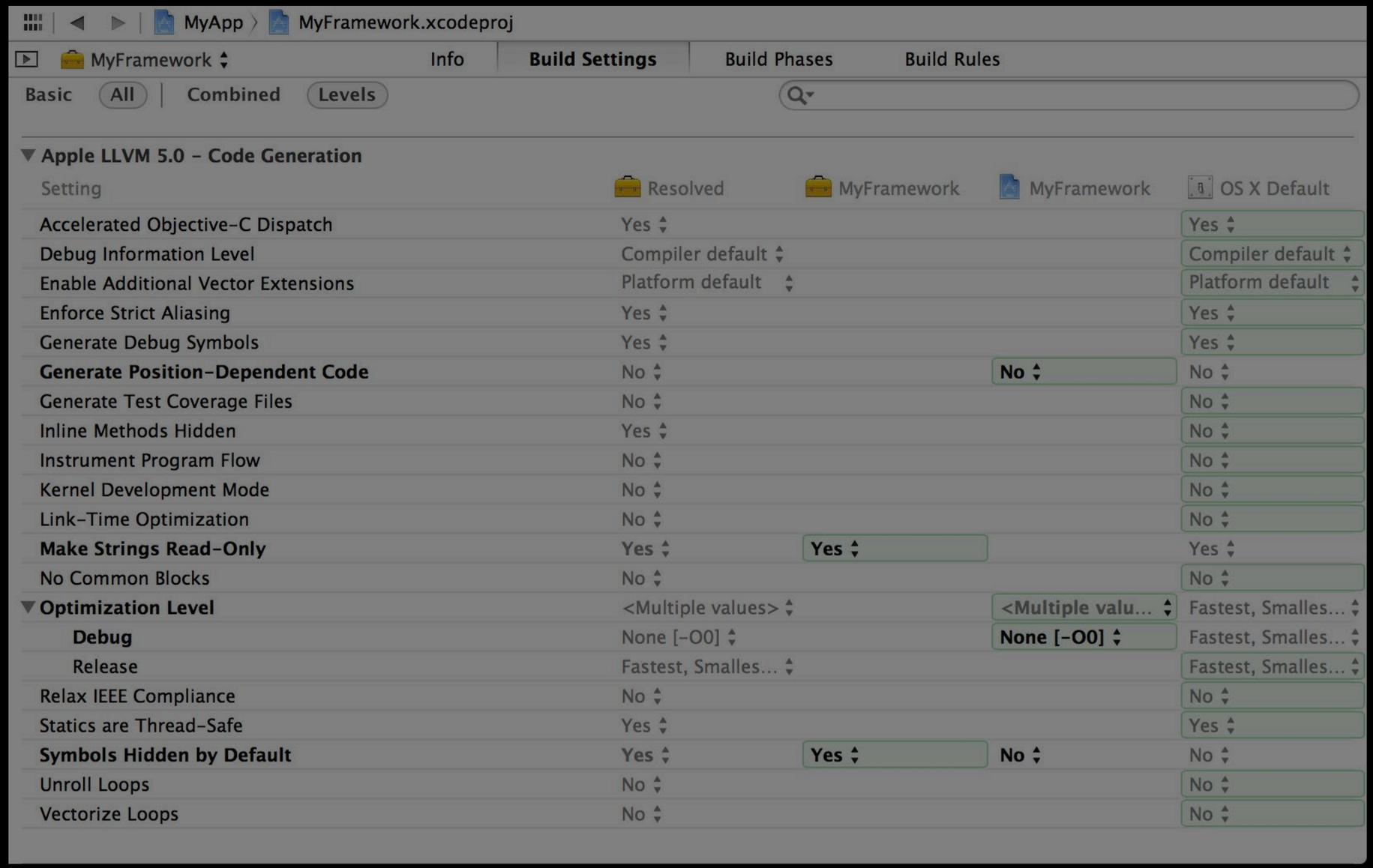
Build Settings



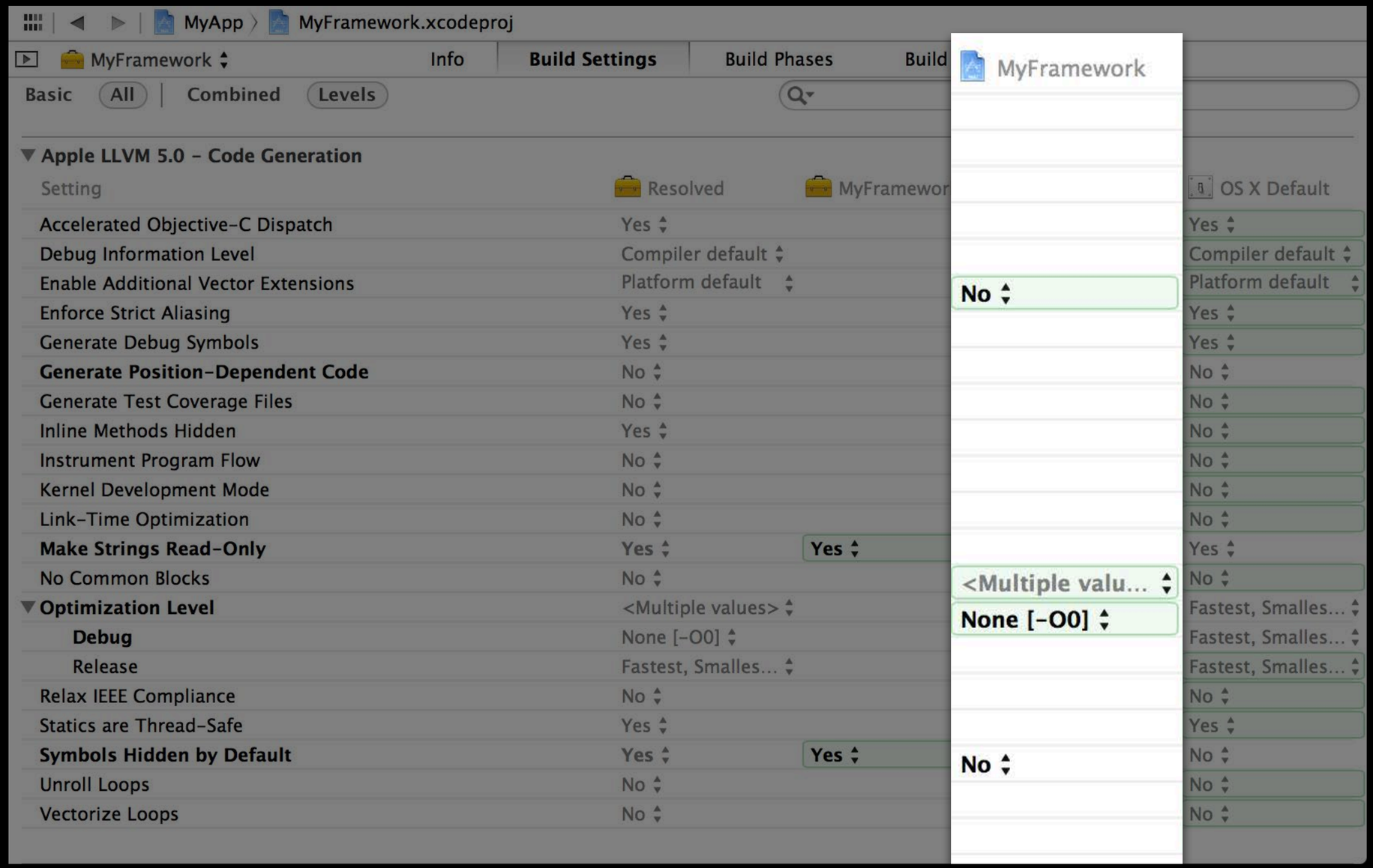
Build Settings



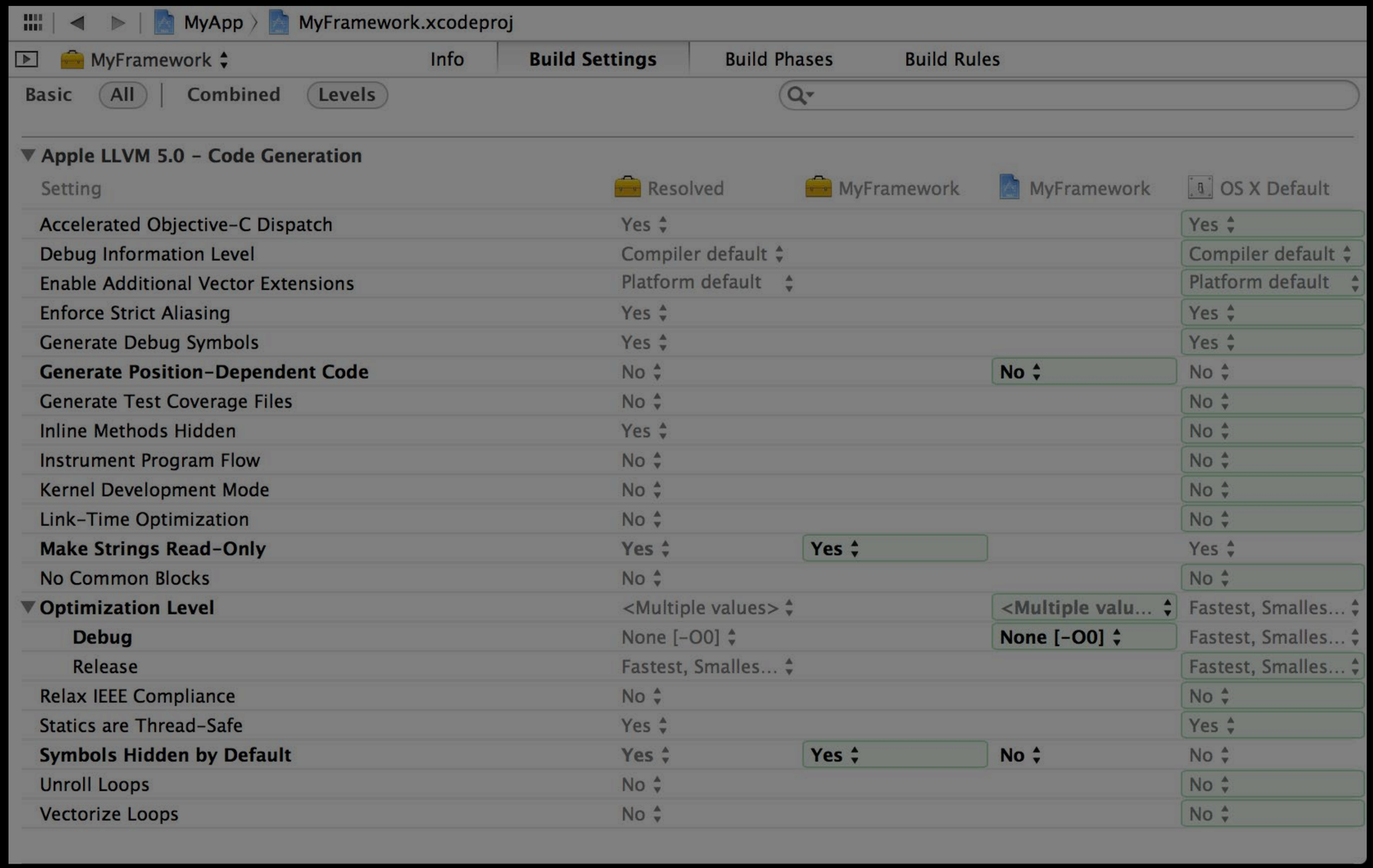
Build Settings



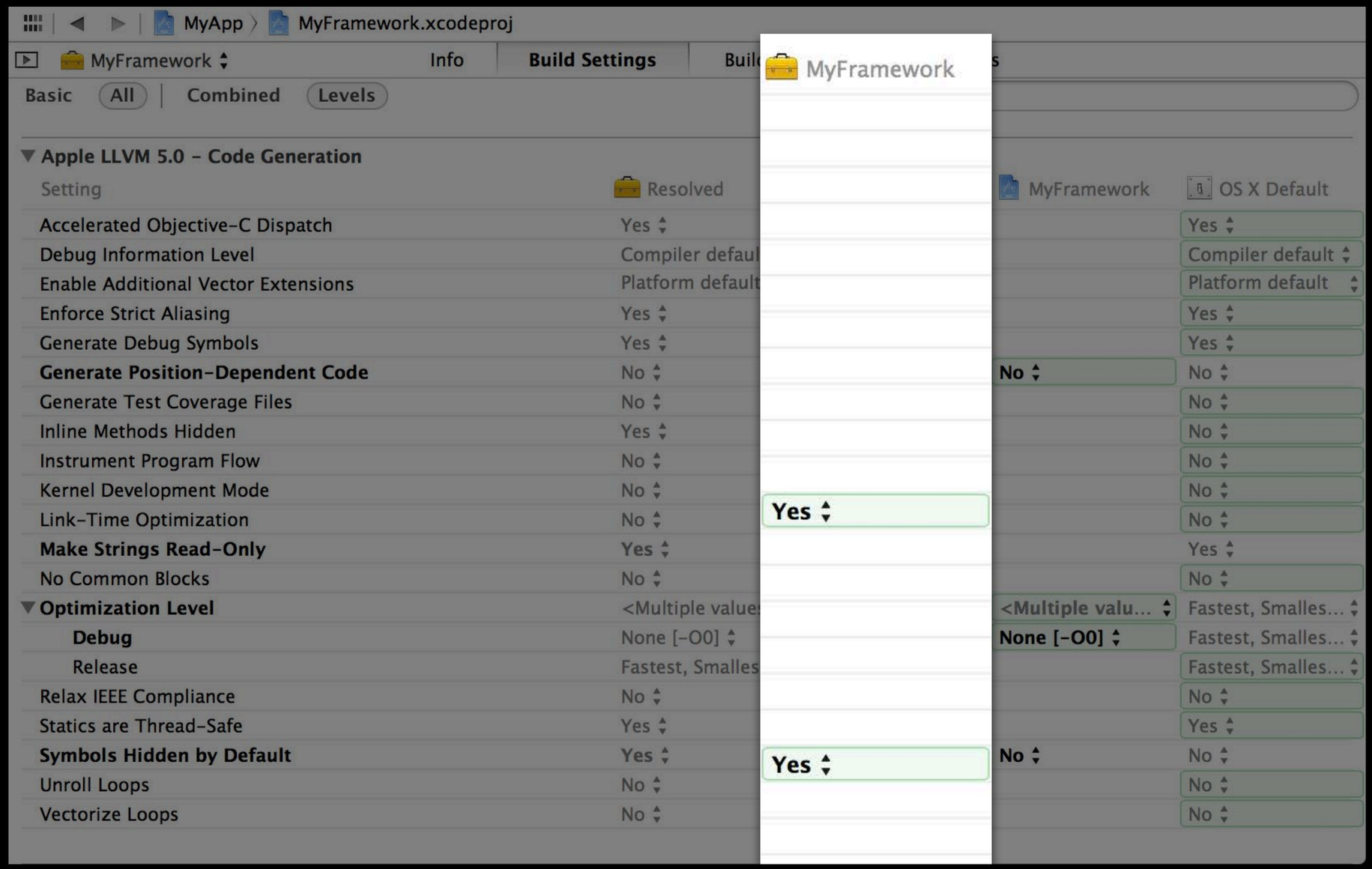
Build Settings



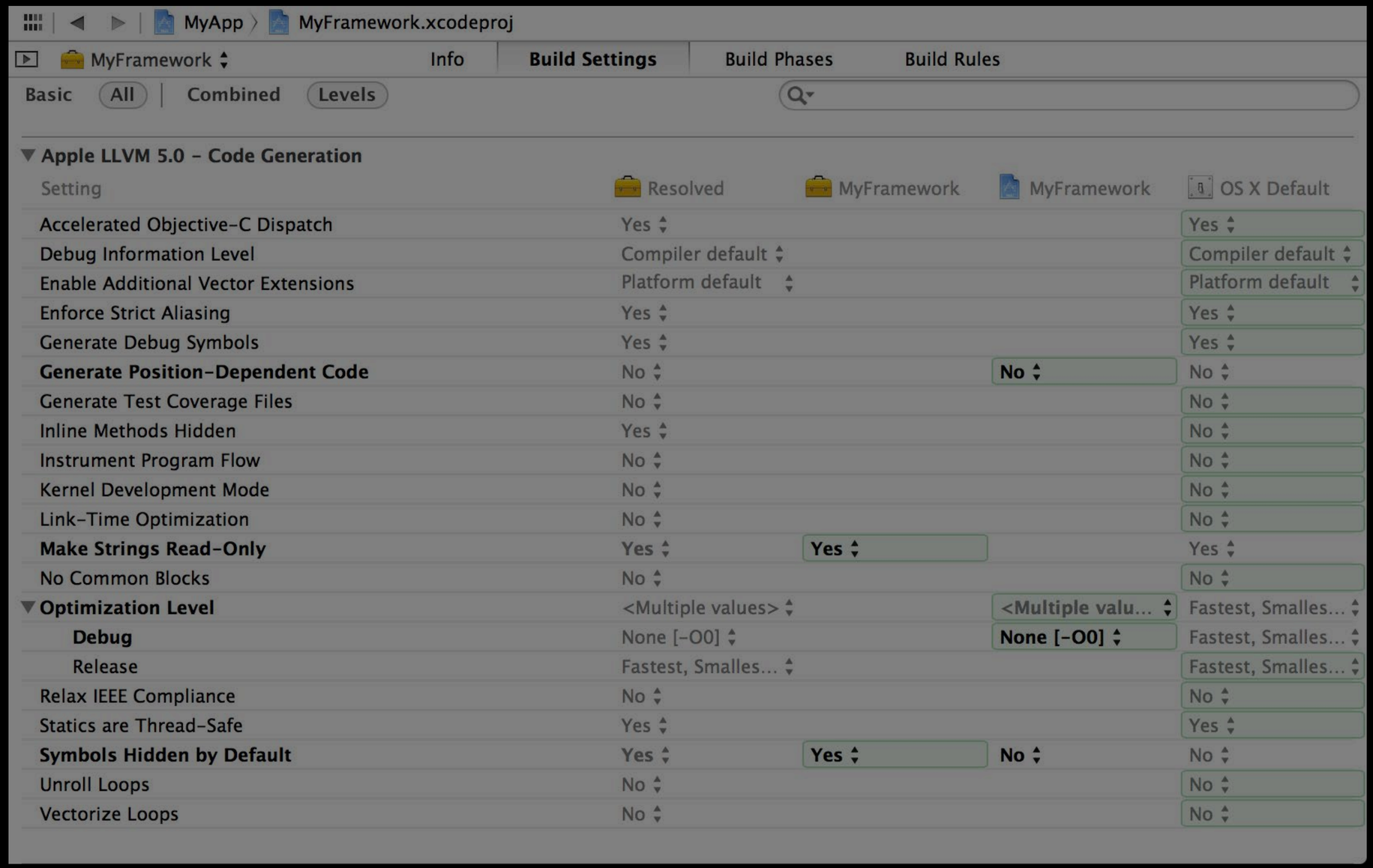
Build Settings



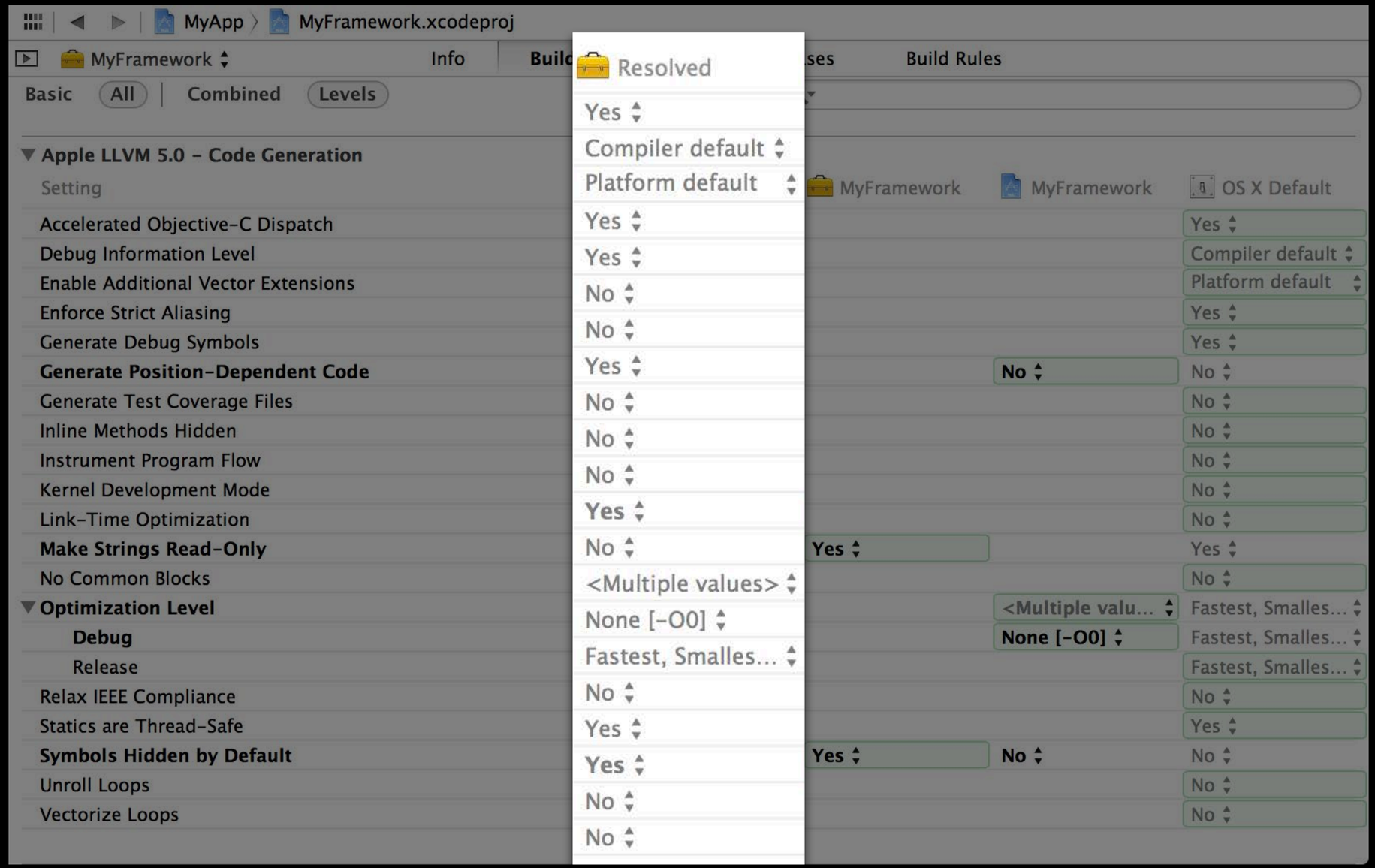
Build Settings



Build Settings



Build Settings

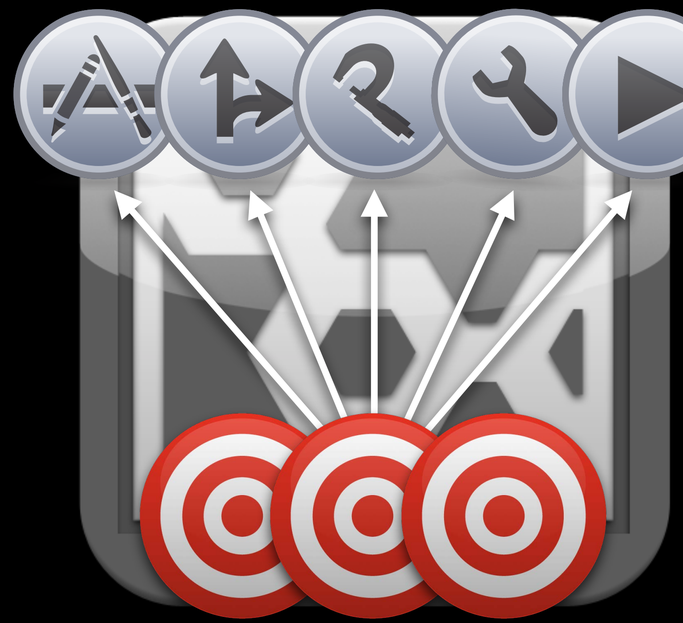


Schemes

Instructions for building targets and performing actions

What's in a Scheme?

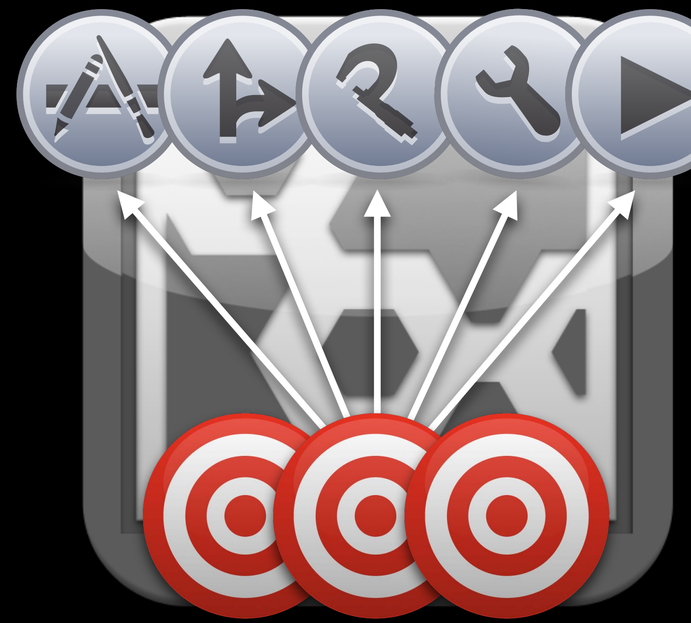
Instructions for building targets and performing actions



What's in a Scheme?

Instructions for building targets and performing actions

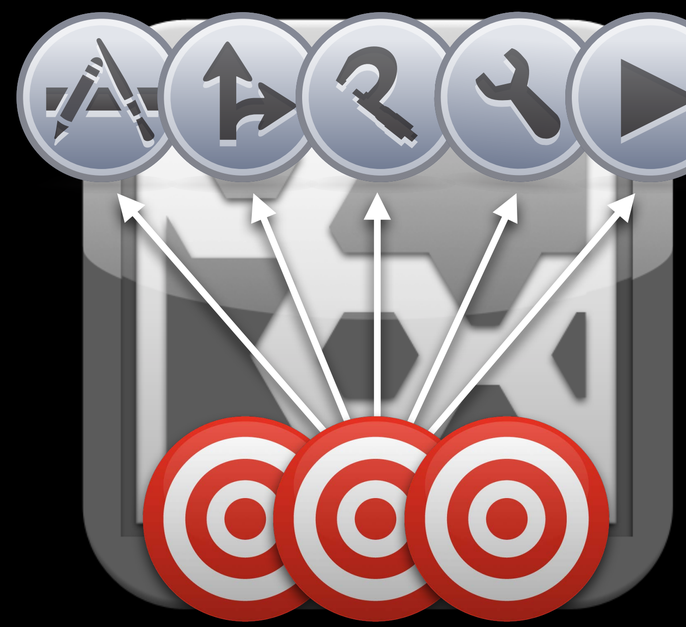
- Actions for running, testing, profiling, analyzing, and archiving products



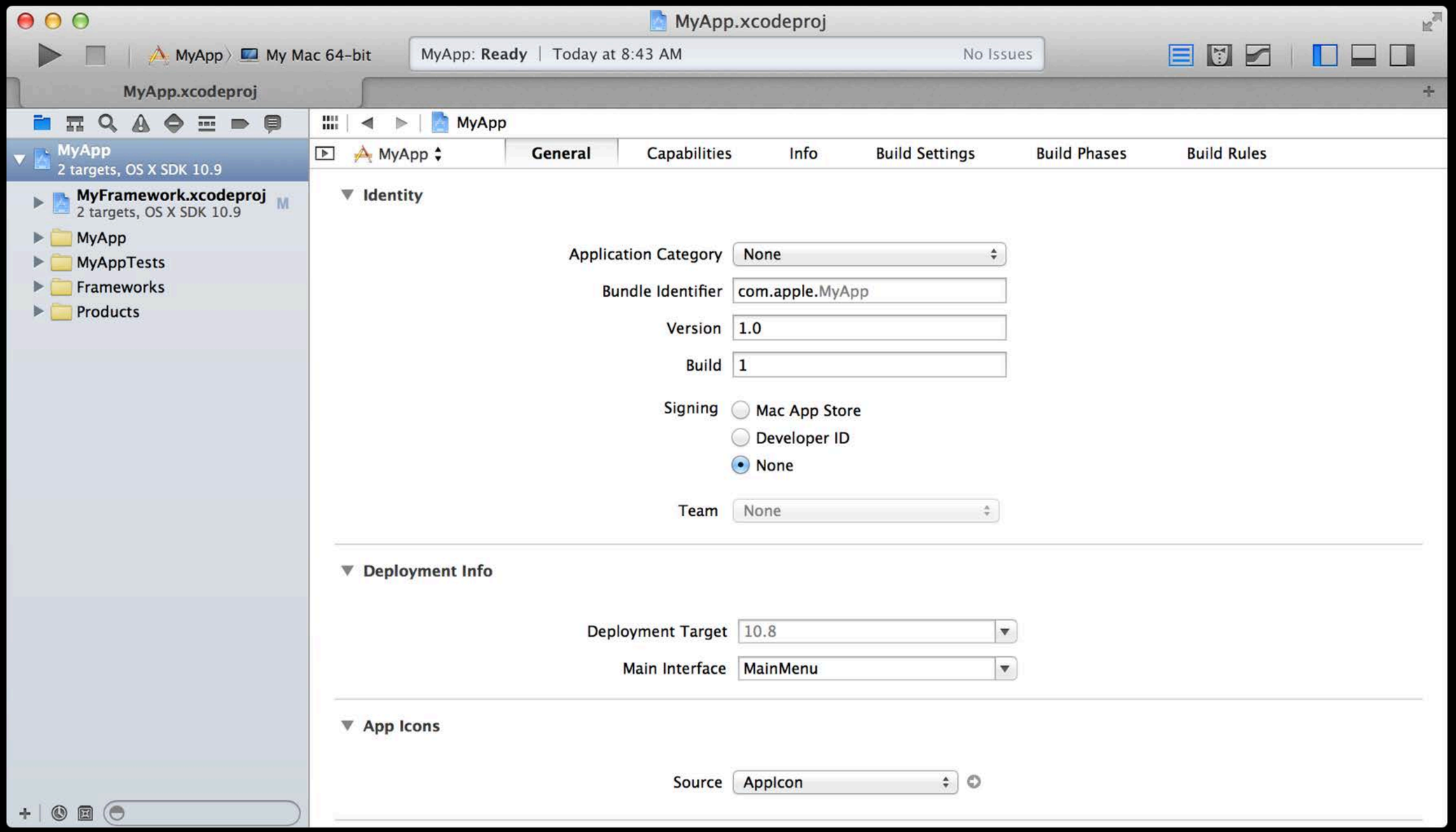
What's in a Scheme?

Instructions for building targets and performing actions

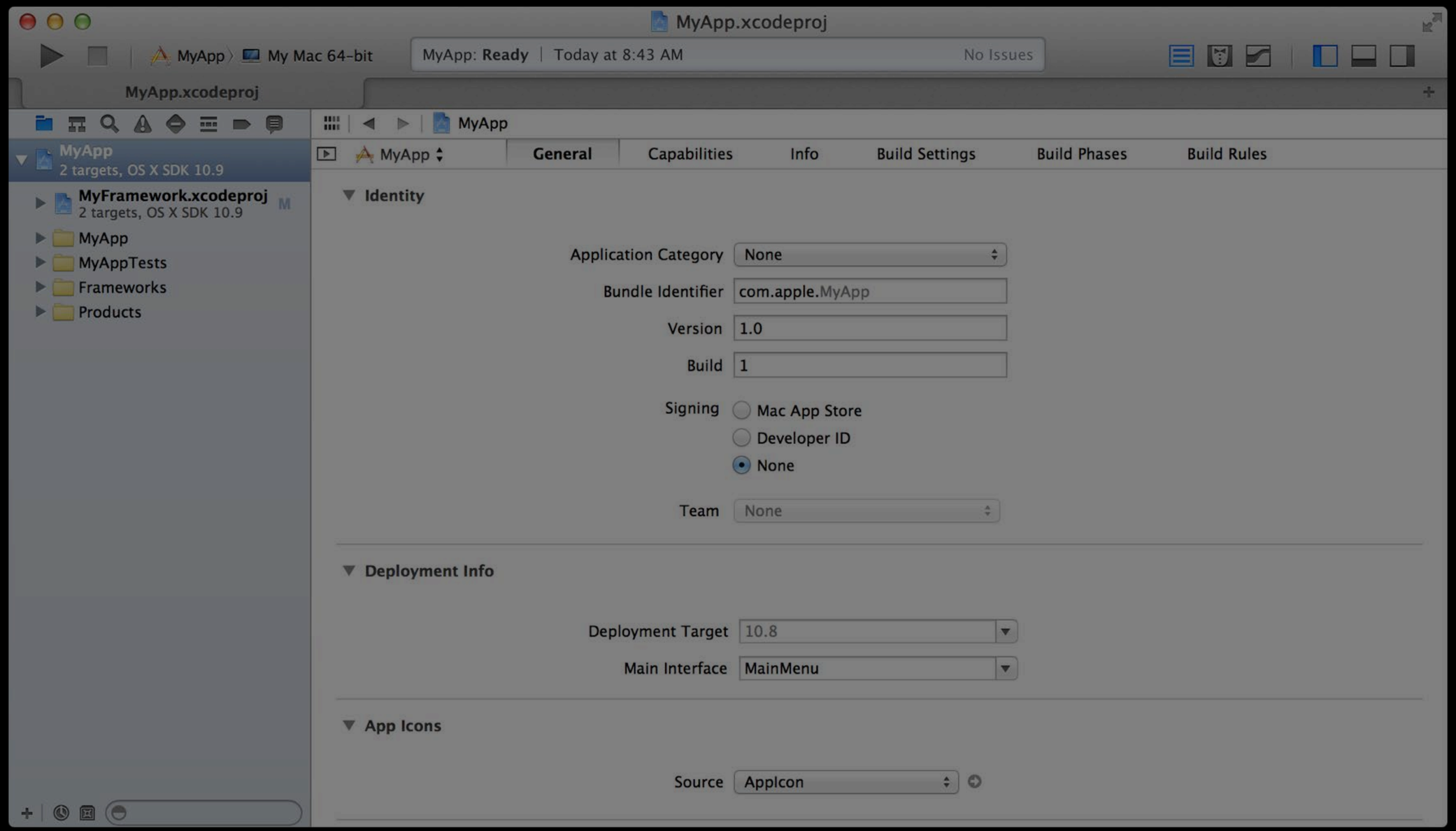
- Actions for running, testing, profiling, analyzing, and archiving products
- A specification of targets to build for each action



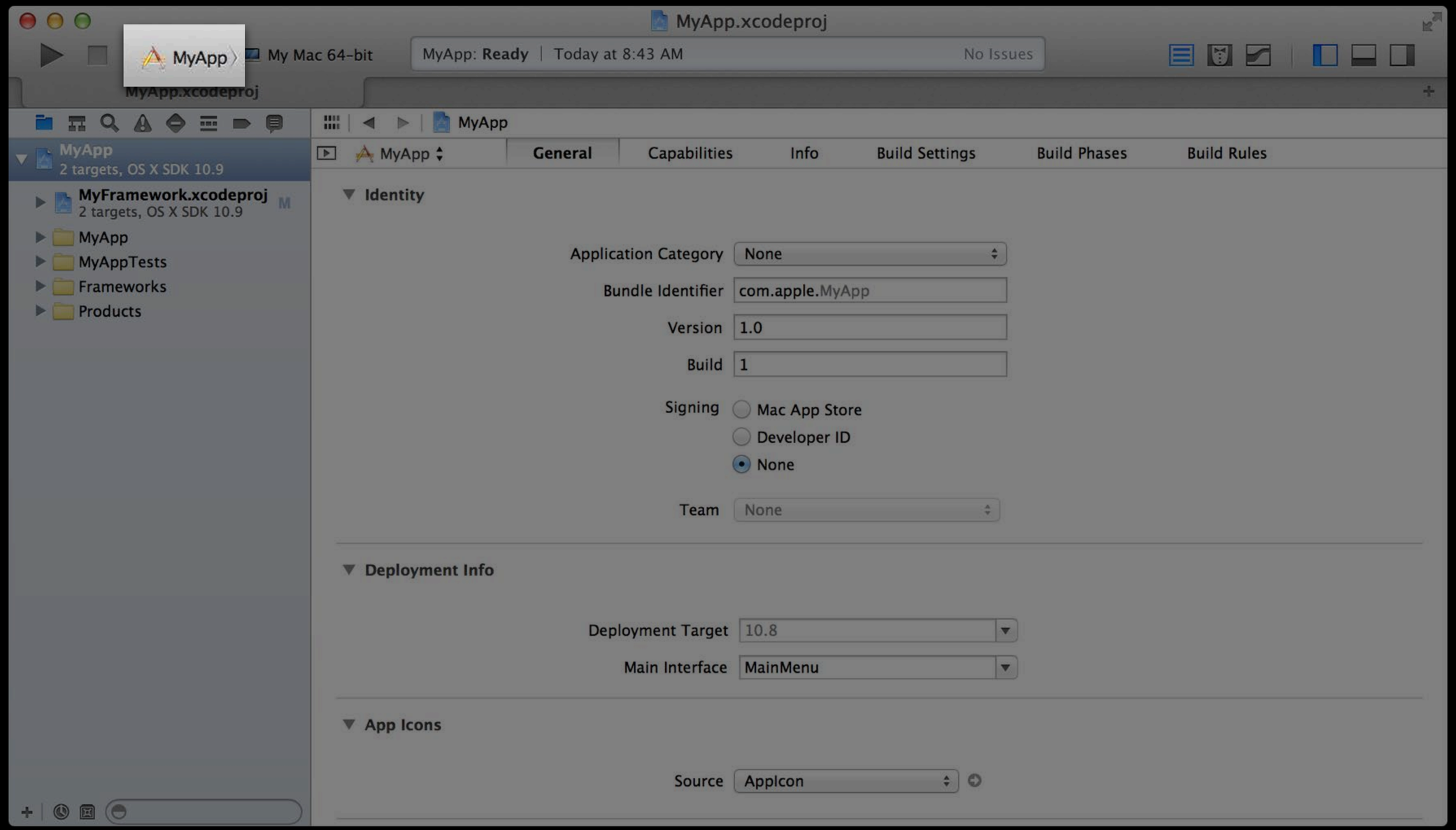
Schemes


MyApp.xcodeproj
MyApp 〉 My Mac 64-bit
MyApp: Ready | Today at 8:43 AM
No Issues
MyApp.xcodeproj
MyApp
MyApp
2 targets, OS X SDK 10.9
MyFramework.xcodeproj
2 targets, OS X SDK 10.9
M
MyApp
MyAppTests
Frameworks
Products
MyApp
General
Capabilities
Info
Build Settings
Build Phases
Build Rules
Identity
Application Category
None
Bundle Identifier
com.apple.MyApp
Version
1.0
Build
1
Signing
Mac App Store
Developer ID
None
Team
None
Deployment Info
Deployment Target
10.8
Main Interface
MainMenu
App Icons
Source
AppIcon

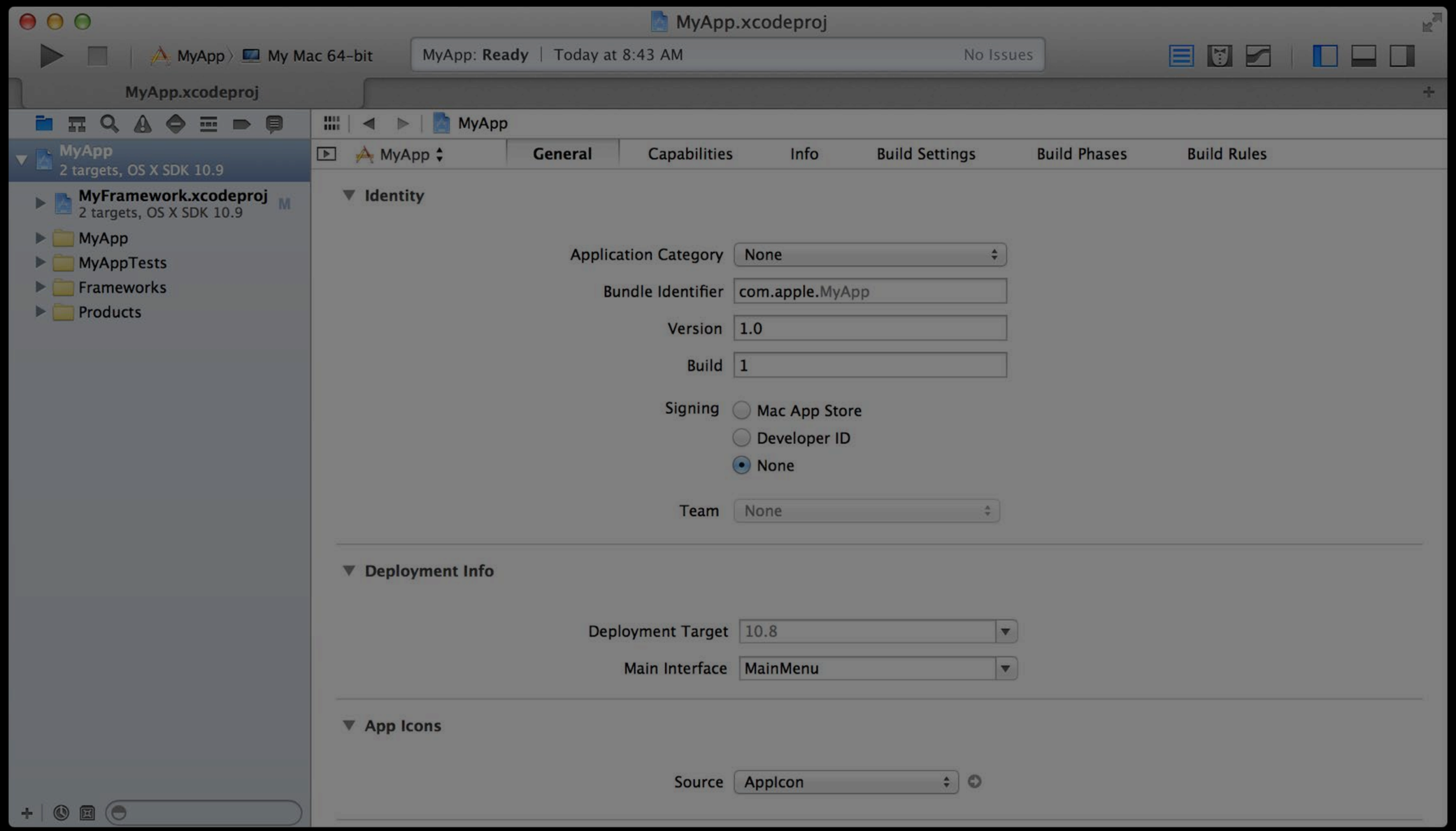
Schemes


MyApp.xcodeproj
MyApp 〉 My Mac 64-bit
MyApp: Ready | Today at 8:43 AM
No Issues
MyApp.xcodeproj
MyApp
MyApp
2 targets, OS X SDK 10.9
MyFramework.xcodeproj
2 targets, OS X SDK 10.9
M
MyApp
MyAppTests
Frameworks
Products
MyApp
General
Capabilities
Info
Build Settings
Build Phases
Build Rules
Identity
Application Category
None
Bundle Identifier
com.apple.MyApp
Version
1.0
Build
1
Signing
Mac App Store
Developer ID
None
Team
None
Deployment Info
Deployment Target
10.8
Main Interface
MainMenu
App Icons
Source
AppIcon

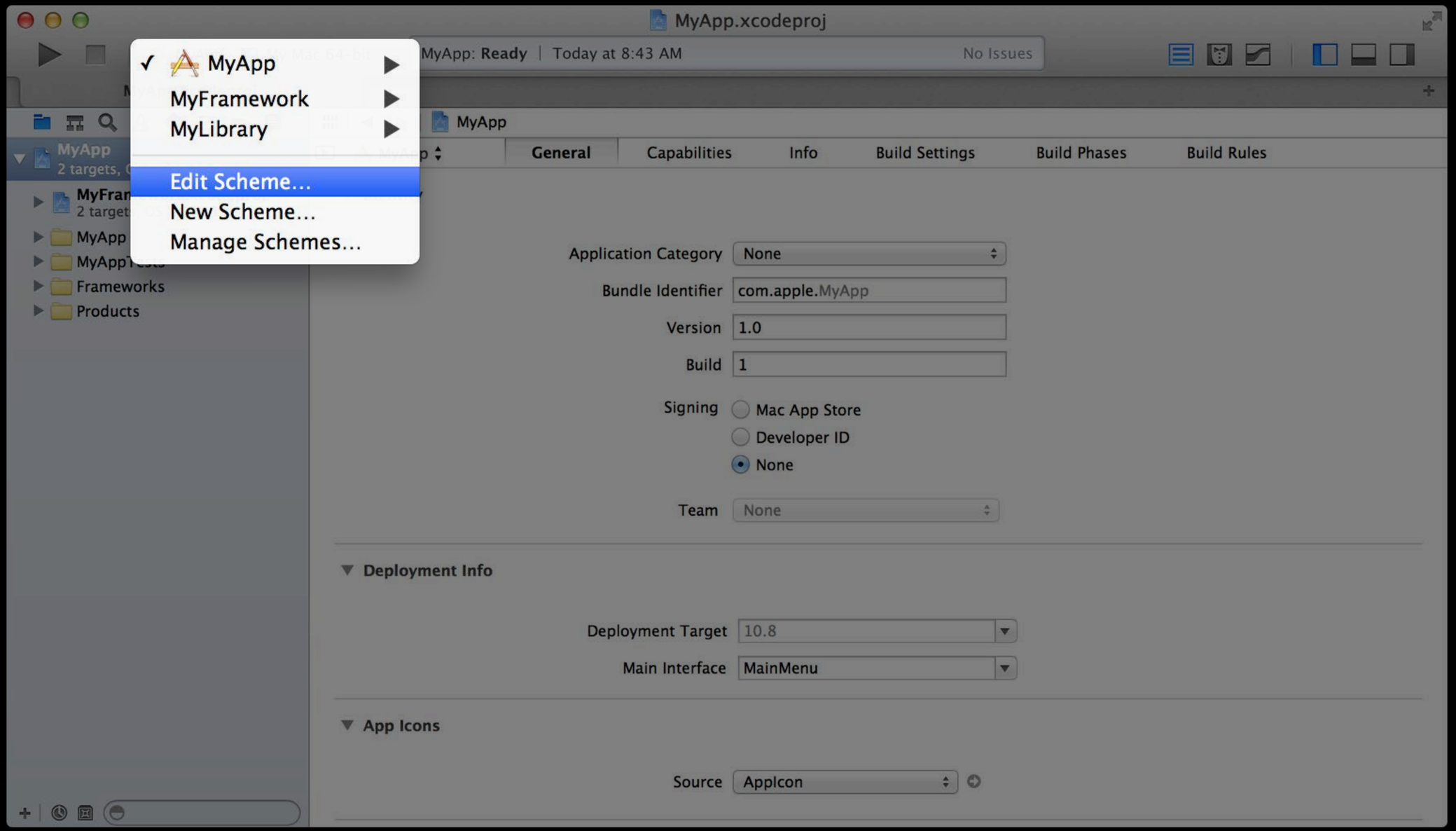
Schemes


MyApp.xcodeproj
MyApp
My Mac 64-bit
MyApp: Ready | Today at 8:43 AM
No Issues
MyApp.xcodeproj
MyApp
MyApp
2 targets, OS X SDK 10.9
MyFramework.xcodeproj
2 targets, OS X SDK 10.9
MyApp
MyAppTests
Frameworks
Products
MyApp
General
Capabilities
Info
Build Settings
Build Phases
Build Rules
Identity
Application Category
None
Bundle Identifier
com.apple.MyApp
Version
1.0
Build
1
Signing
Mac App Store
Developer ID
None
Team
None
Deployment Info
Deployment Target
10.8
Main Interface
MainMenu
App Icons
Source
AppIcon

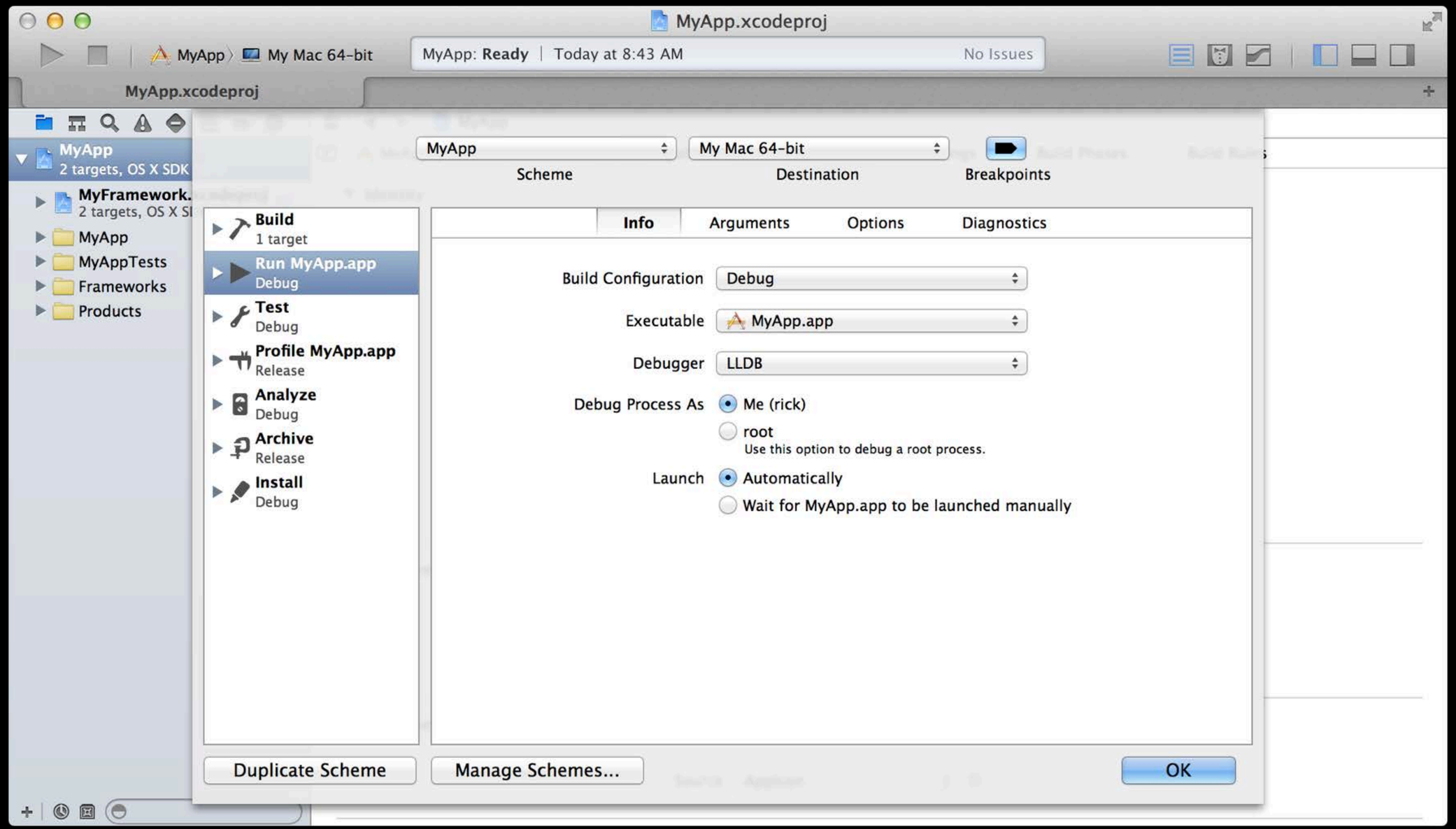
Schemes


MyApp.xcodeproj
MyApp 〉 My Mac 64-bit
MyApp: Ready | Today at 8:43 AM
No Issues
MyApp.xcodeproj
MyApp
MyApp
2 targets, OS X SDK 10.9
MyFramework.xcodeproj
2 targets, OS X SDK 10.9
M
MyApp
MyAppTests
Frameworks
Products
MyApp
General
Capabilities
Info
Build Settings
Build Phases
Build Rules
Identity
Application Category
None
Bundle Identifier
com.apple.MyApp
Version
1.0
Build
1
Signing
Mac App Store
Developer ID
None
Team
None
Deployment Info
Deployment Target
10.8
Main Interface
MainMenu
App Icons
Source
AppIcon

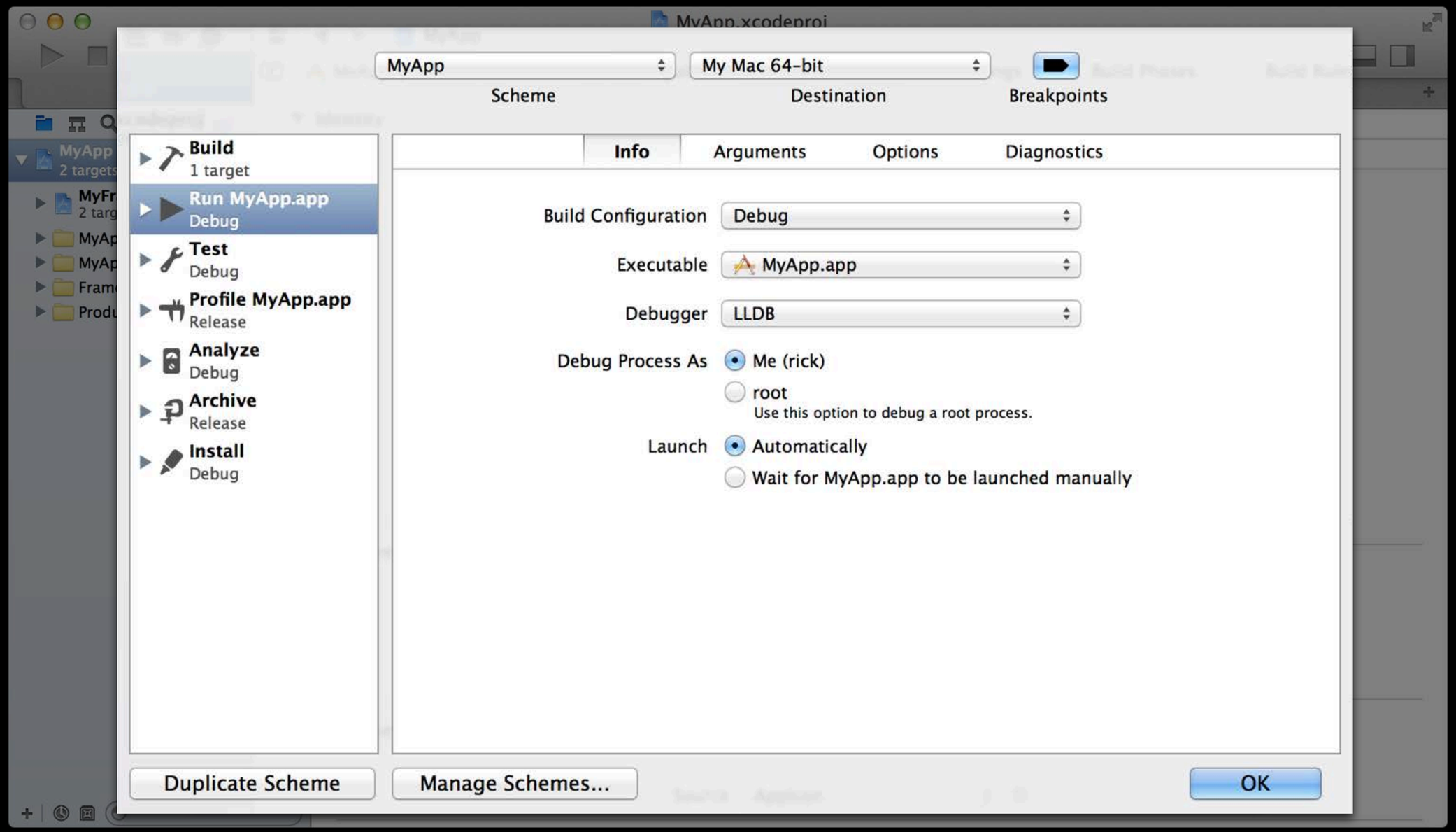
Schemes


MyApp.xcodeproj
MyApp: Ready | Today at 8:43 AM
No Issues
MyApp
MyFramework
MyLibrary
Edit Scheme…
New Scheme…
Manage Schemes…
MyApp
2 targets,
MyFran
2 target
MyApp
MyAppTests
Frameworks
Products
MyApp
General
Capabilities
Info
Build Settings
Build Phases
Build Rules
Application Category
None
Bundle Identifier
com.apple.MyApp
Version
1.0
Build
1
Signing
Mac App Store
Developer ID
None
Team
None
Deployment Info
Deployment Target
10.8
Main Interface
MainMenu
App Icons
Source
AppIcon

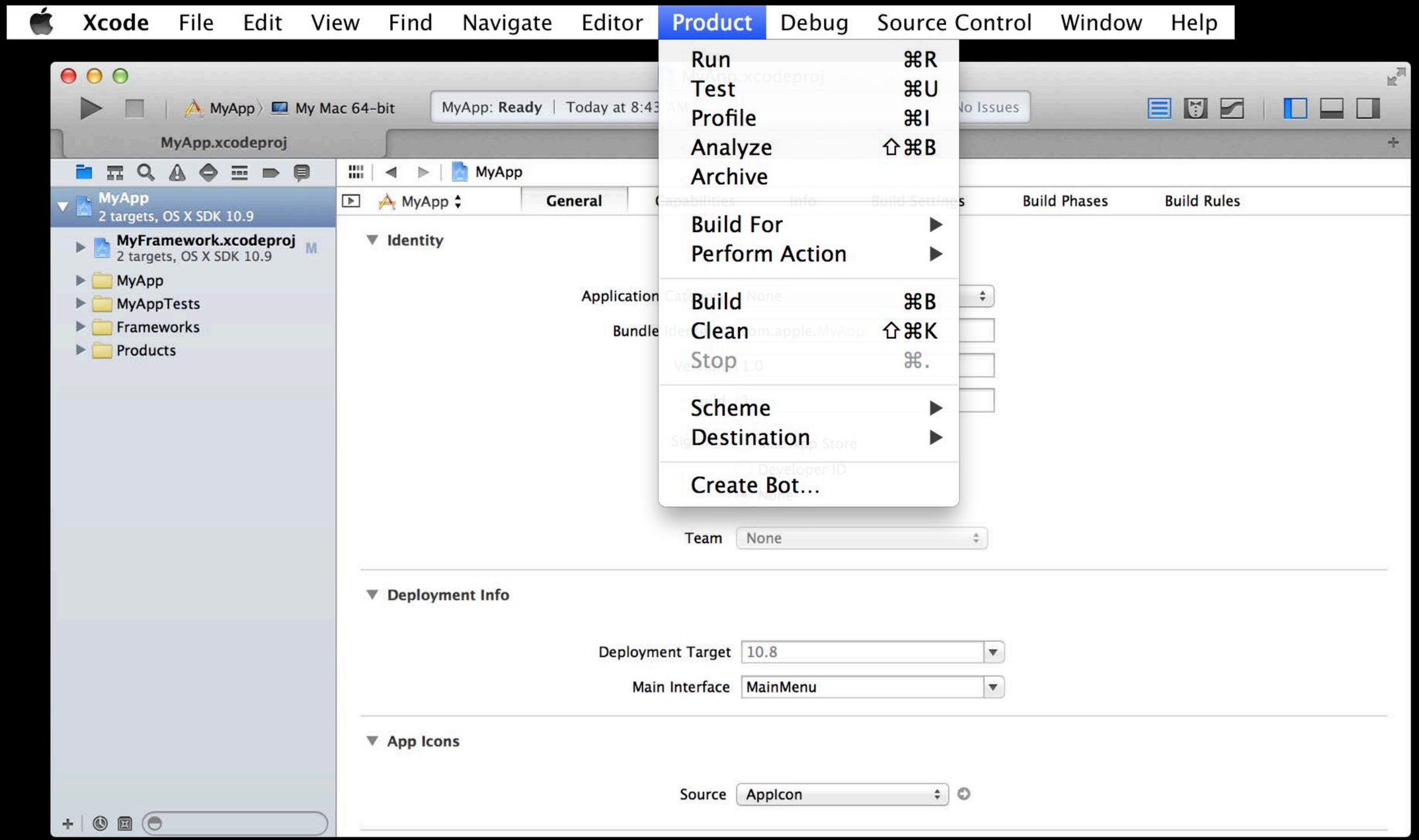
Schemes


MyApp.xcodeproj
MyApp 〉 My Mac 64-bit
MyApp: Ready | Today at 8:43 AM
No Issues
MyApp.xcodeproj
MyApp
2 targets, OS X SDK
MyFramework.
2 targets, OS X SI
MyApp
MyAppTests
Frameworks
Products
MyApp
Scheme
My Mac 64-bit
Destination
Breakpoints
Build
1 target
Run MyApp.app
Debug
Test
Debug
Profile MyApp.app
Release
Analyze
Debug
Archive
Release
Install
Debug
Info
Arguments
Options
Diagnostics
Build Configuration
Debug
Executable
MyApp.app
Debugger
LLDB
Debug Process As
Me (rick)
root
Use this option to debug a root process.
Launch
Automatically
Wait for MyApp.app to be launched manually
Duplicate Scheme
Manage Schemes...
OK

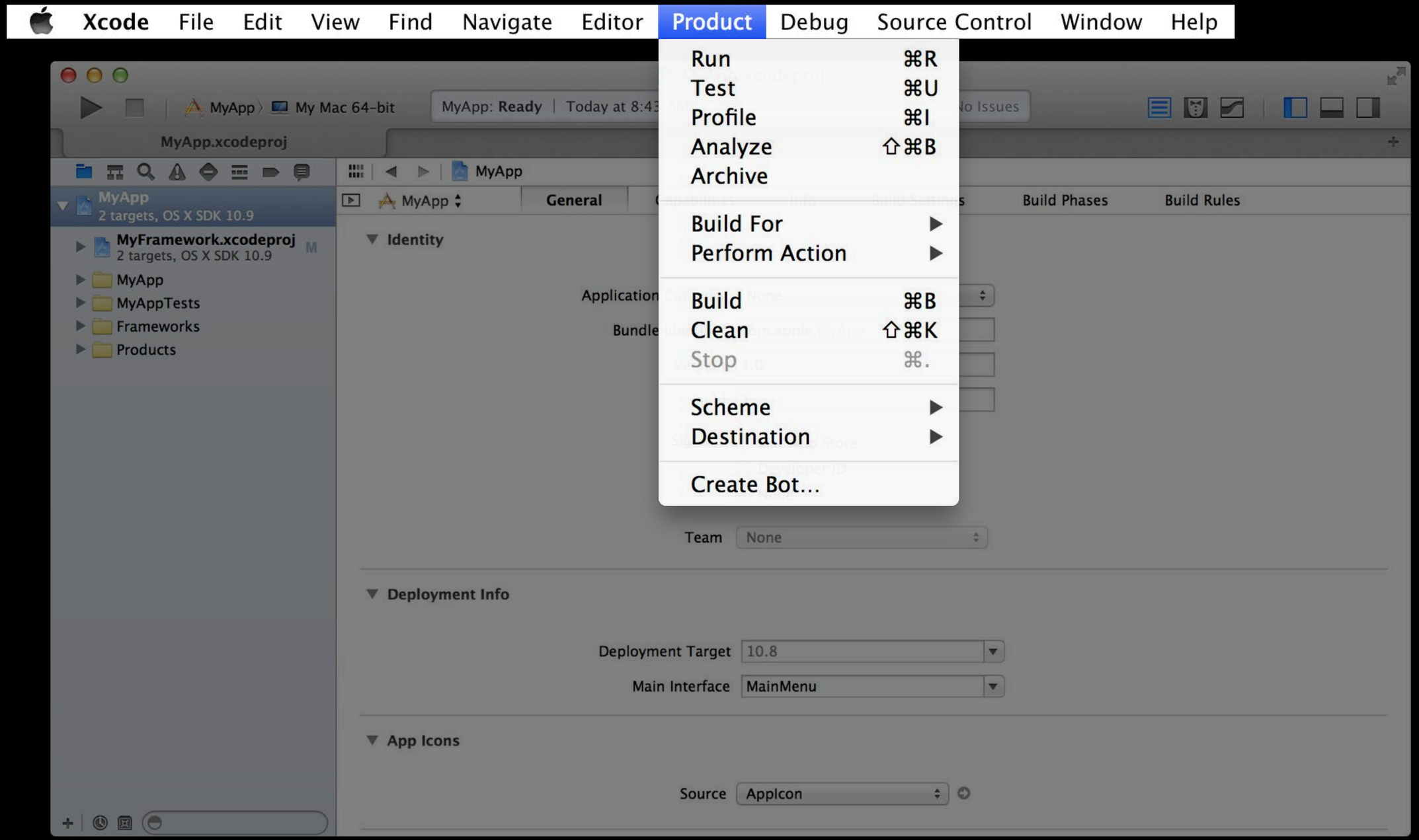
Schemes


MyApp.xcodeproj
MyApp
My Mac 64-bit
Scheme
Destination
Breakpoints
Build
1 target
Run MyApp.app
Debug
Test
Debug
Profile MyApp.app
Release
Analyze
Debug
Archive
Release
Install
Debug
Info
Arguments
Options
Diagnostics
Build Configuration
Debug
Executable
MyApp.app
Debugger
LLDB
Debug Process As
Me (rick)
root
Use this option to debug a root process.
Launch
Automatically
Wait for MyApp.app to be launched manually
Duplicate Scheme
Manage Schemes...
OK

Schemes


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Run ⌘R
Test ⌘U
Profile ⌘I
Analyze ⇧⌘B
Archive
Build For
Perform Action
Build ⌘B
Clean ⇧⌘K
Stop ⌘.
Scheme
Destination
Create Bot…
MyApp My Mac 64-bit
MyApp: Ready | Today at 8:43
MyApp.xcodeproj
MyApp
2 targets, OS X SDK 10.9
MyFramework.xcodeproj
2 targets, OS X SDK 10.9
MyApp
MyAppTests
Frameworks
Products
General
Build Phases
Build Rules
Identity
Team None
Deployment Info
Deployment Target 10.8
Main Interface MainMenu
App Icons
Source AppIcon

Schemes


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Run ⌘R
Test ⌘U
Profile ⌘I
Analyze ⇧⌘B
Archive
Build For
Perform Action
Build ⌘B
Clean ⇧⌘K
Stop ⌘.
Scheme
Destination
Create Bot…
MyApp
My Mac 64-bit
MyApp: Ready
MyApp.xcodeproj
MyApp
2 targets, OS X SDK 10.9
MyFramework.xcodeproj
2 targets, OS X SDK 10.9
MyApp
MyAppTests
Frameworks
Products
General
Build Phases
Build Rules
Identity
Team
None
Deployment Info
Deployment Target
10.8
Main Interface
MainMenu
App Icons
Source
AppIcon

When are Schemes Created?

When are Schemes Created?

- Automatically:
  - When creating a new target or project
  - When opening a project or workspace for the first time

When are Schemes Created?

- Automatically:
  - When creating a new target or project
  - When opening a project or workspace for the first time
- Manually by the user

Why Edit a Scheme?

Why Edit a Scheme?

- To change:

Why Edit a Scheme?

- To change:
  - The build configuration to use for each action

Why Edit a Scheme?

- To change:
  - The build configuration to use for each action
  - The unit tests to run

Why Edit a Scheme?

- To change:
  - The build configuration to use for each action
  - The unit tests to run
  - The arguments and environment for your app

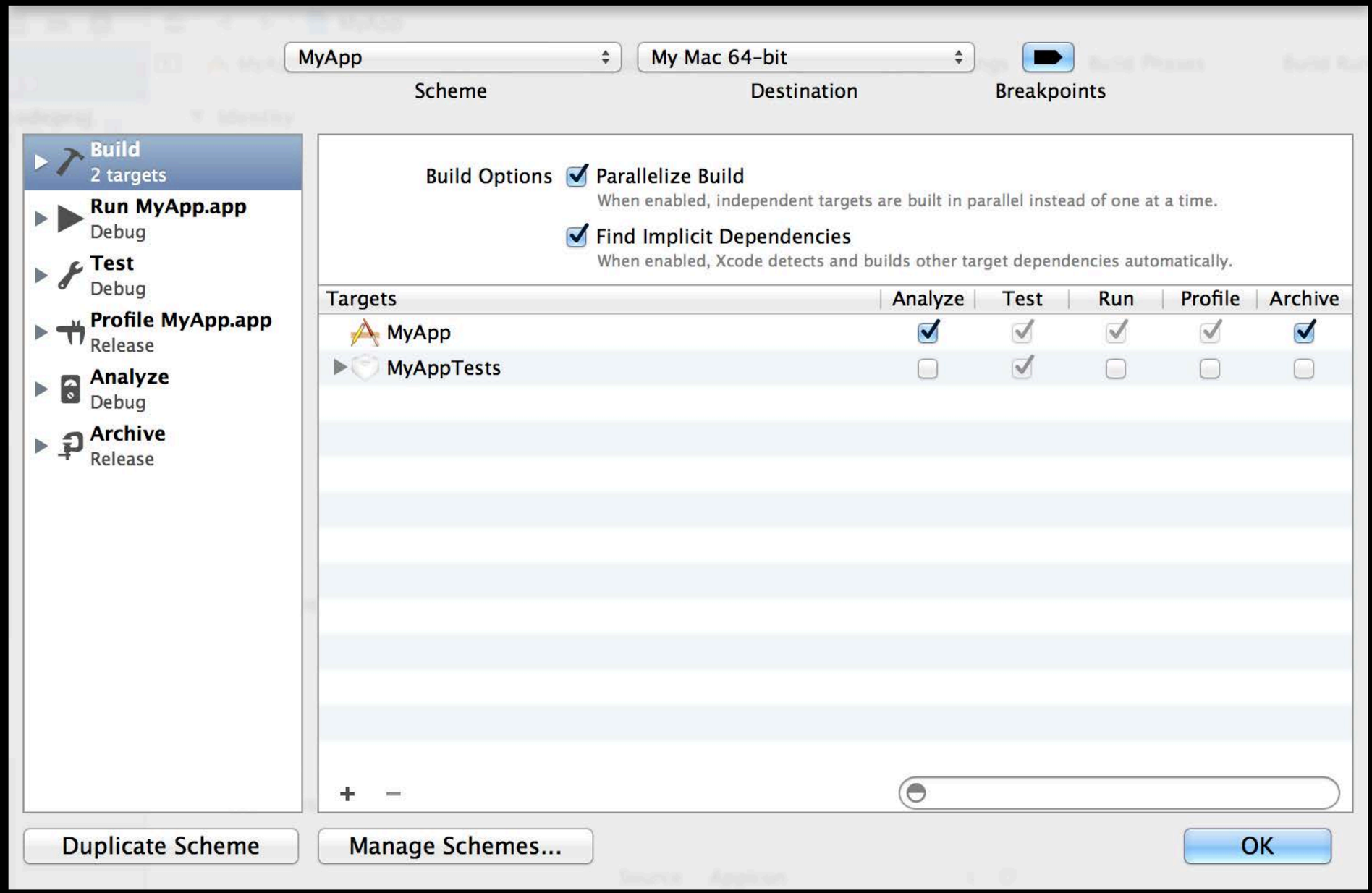
Why Edit a Scheme?

- To change:
  - The build configuration to use for each action
  - The unit tests to run
  - The arguments and environment for your app
  - Diagnostics enabled in the debugger

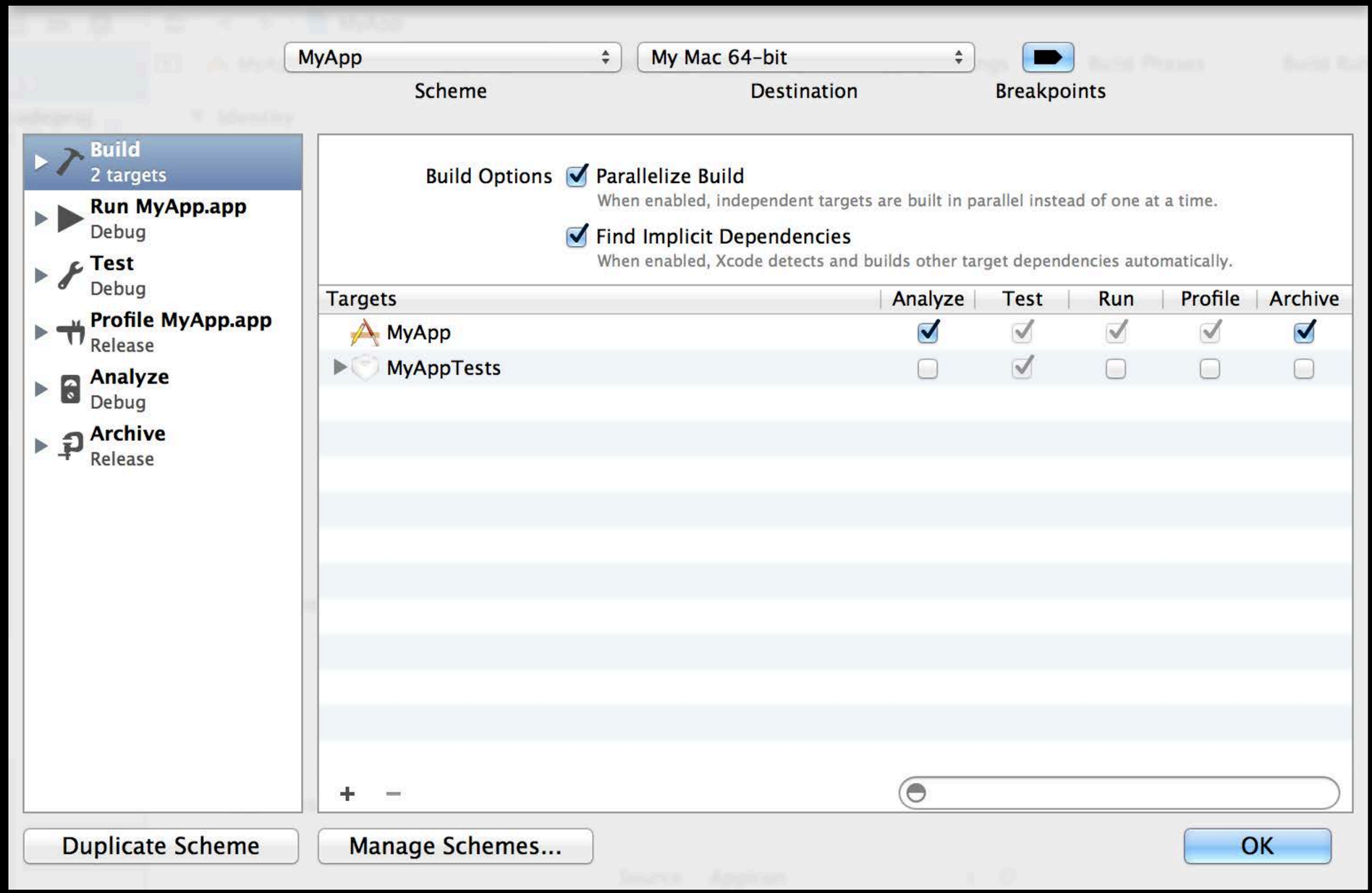
Why Edit a Scheme?

- To change:
  - The build configuration to use for each action
  - The unit tests to run
  - The arguments and environment for your app
  - Diagnostics enabled in the debugger
  - Other options

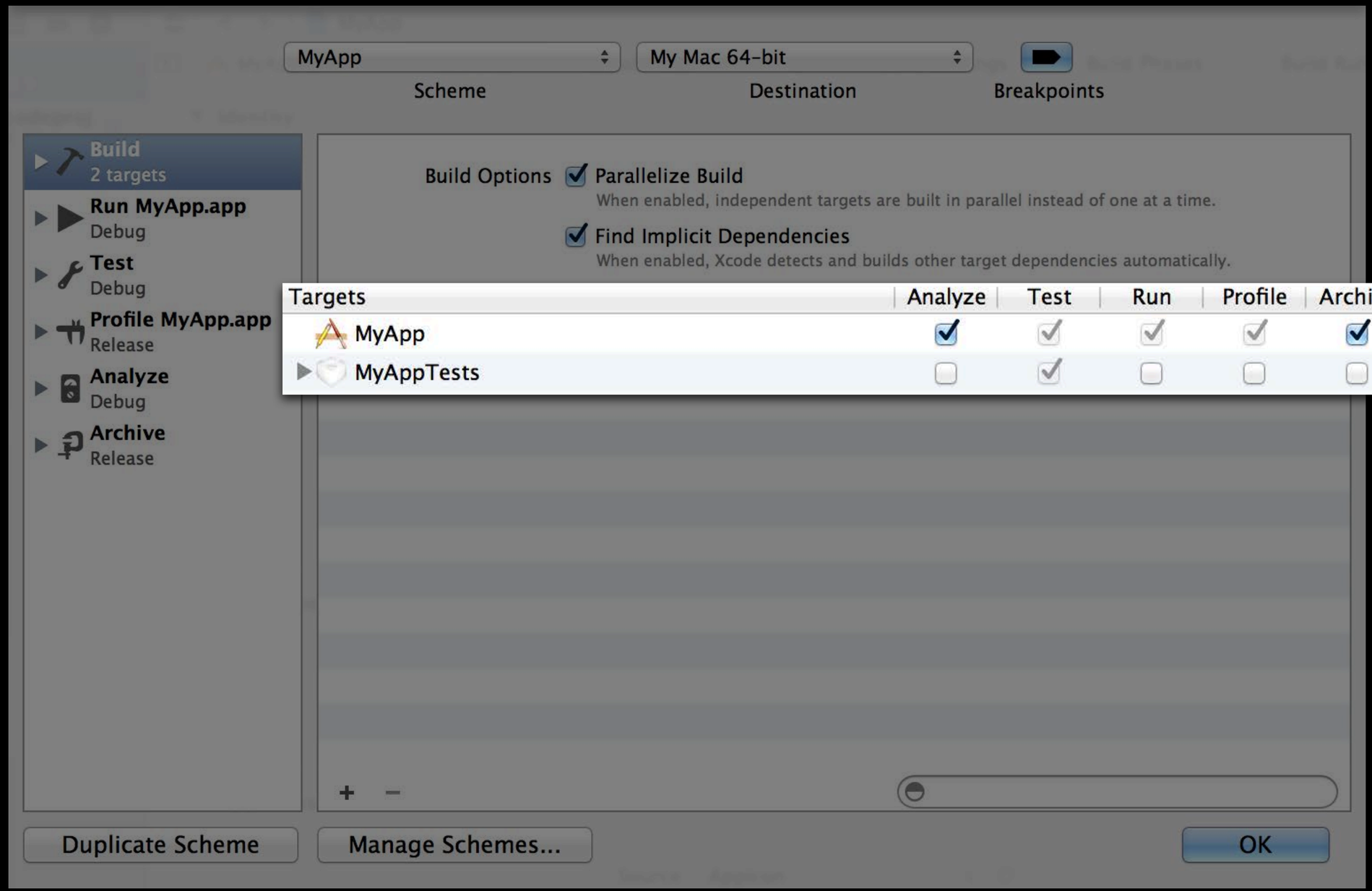
Building


MyApp
My Mac 64-bit
Scheme
Destination
Breakpoints
Build
2 targets
Run MyApp.app
Debug
Test
Debug
Profile MyApp.app
Release
Analyze
Debug
Archive
Release
Build Options
Parallelize Build
When enabled, independent targets are built in parallel instead of one at a time.
Find Implicit Dependencies
When enabled, Xcode detects and builds other target dependencies automatically.
Targets
Analyze
Test
Run
Profile
Archive
MyApp
MyAppTests
Duplicate Scheme
Manage Schemes...
OK

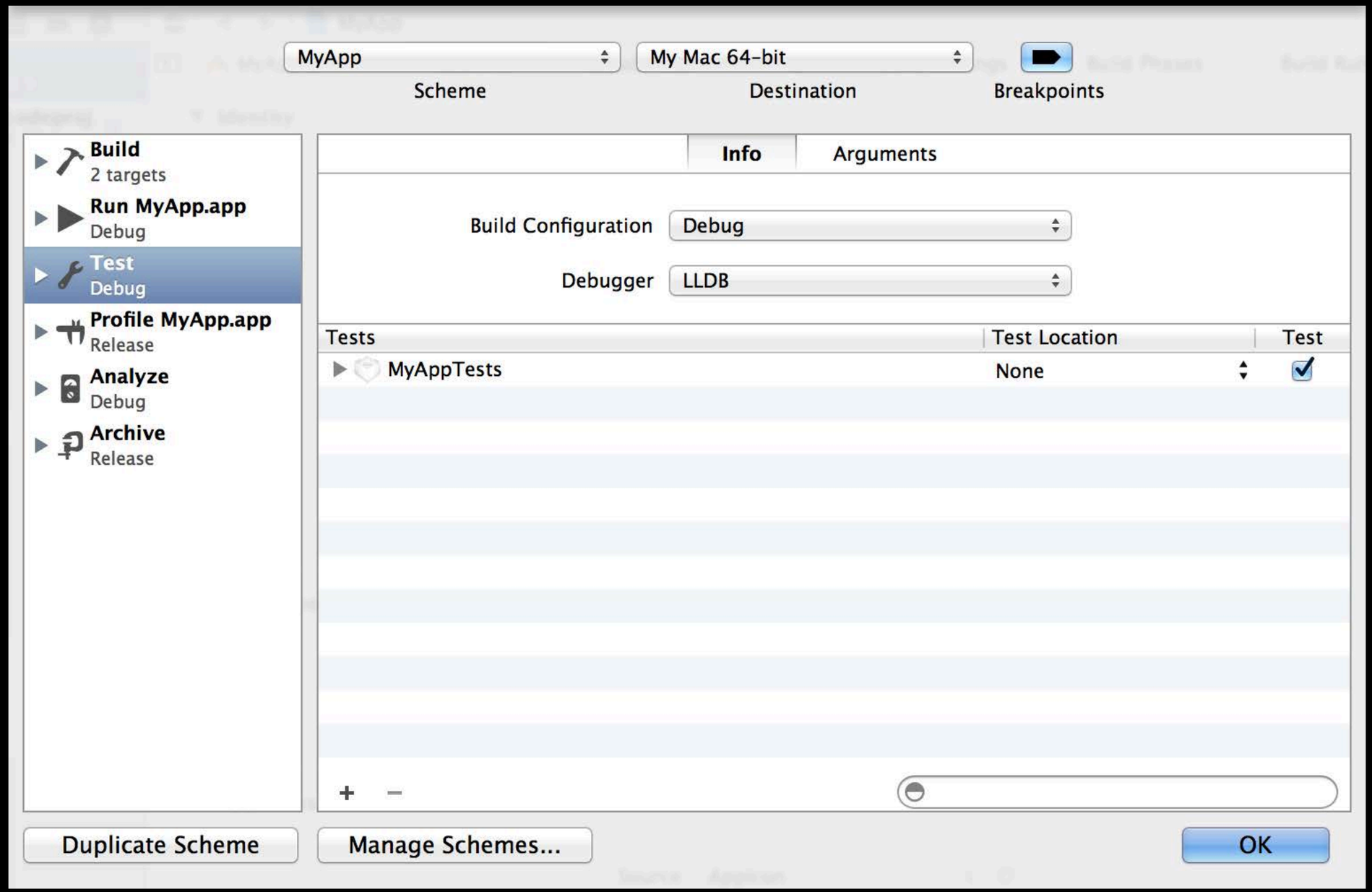
Building


MyApp
My Mac 64-bit
Scheme
Destination
Breakpoints
Build
2 targets
Run MyApp.app
Debug
Test
Debug
Profile MyApp.app
Release
Analyze
Debug
Archive
Release
Build Options
Parallelize Build
When enabled, independent targets are built in parallel instead of one at a time.
Find Implicit Dependencies
When enabled, Xcode detects and builds other target dependencies automatically.
Targets
Analyze
Test
Run
Profile
Archive
MyApp
MyAppTests
Duplicate Scheme
Manage Schemes...
OK

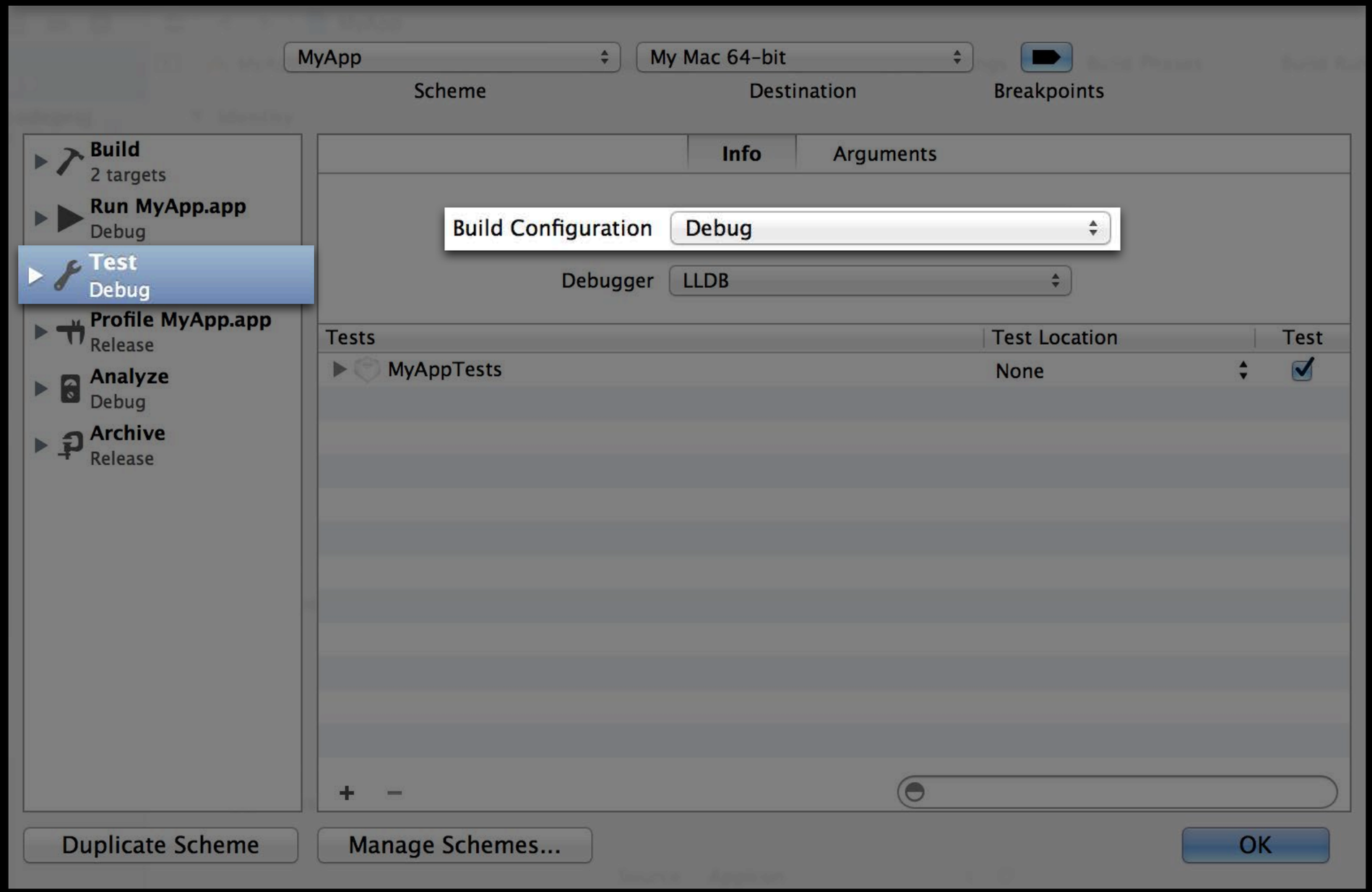
Building


MyApp
My Mac 64-bit
Scheme
Destination
Breakpoints
Build
2 targets
Run MyApp.app
Debug
Test
Debug
Profile MyApp.app
Release
Analyze
Debug
Archive
Release
Build Options
Parallelize Build
When enabled, independent targets are built in parallel instead of one at a time.
Find Implicit Dependencies
When enabled, Xcode detects and builds other target dependencies automatically.
Targets
Analyze
Test
Run
Profile
Archive
MyApp
MyAppTests
Duplicate Scheme
Manage Schemes...
OK

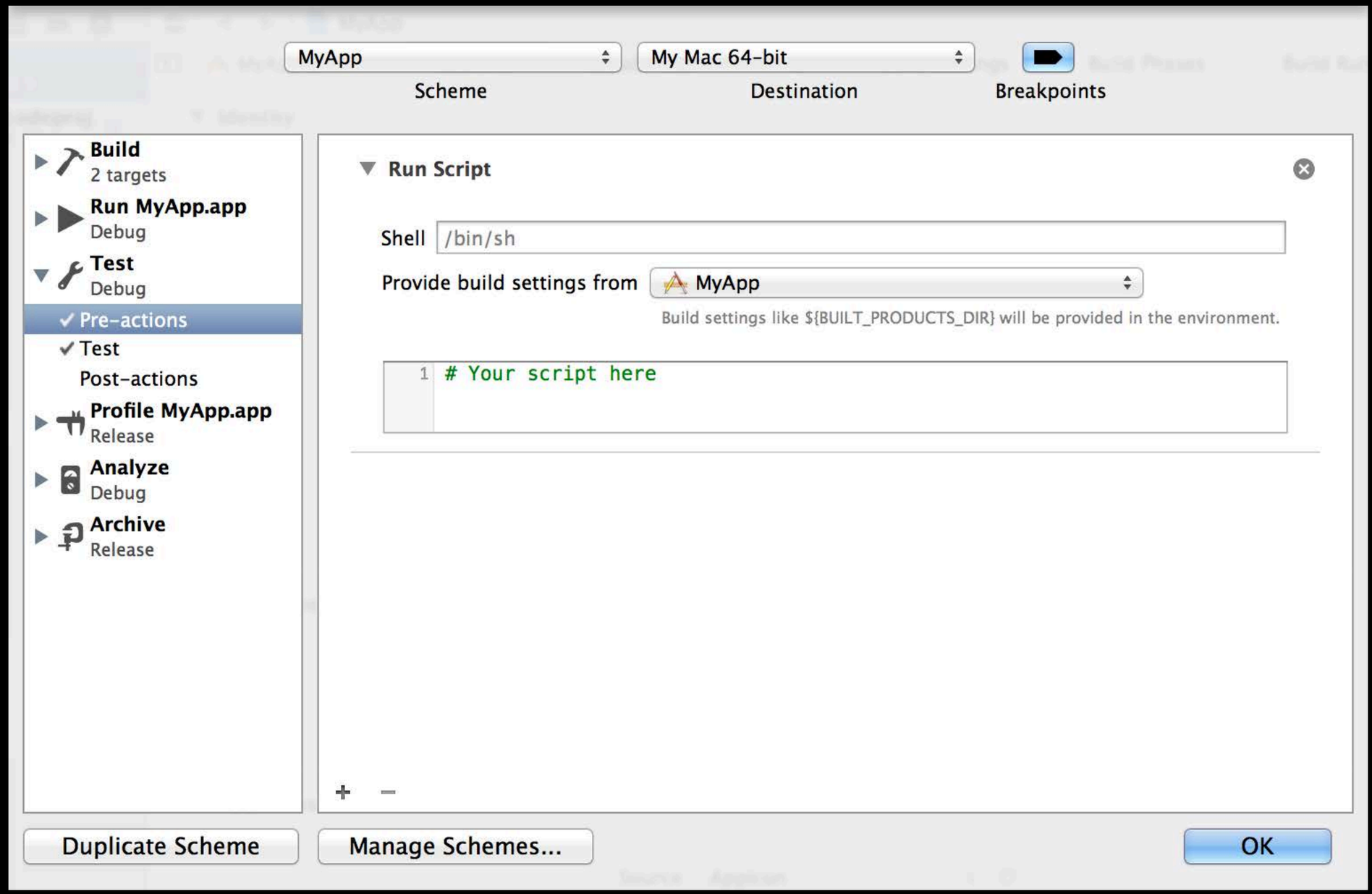
Scheme Action Build Configuration



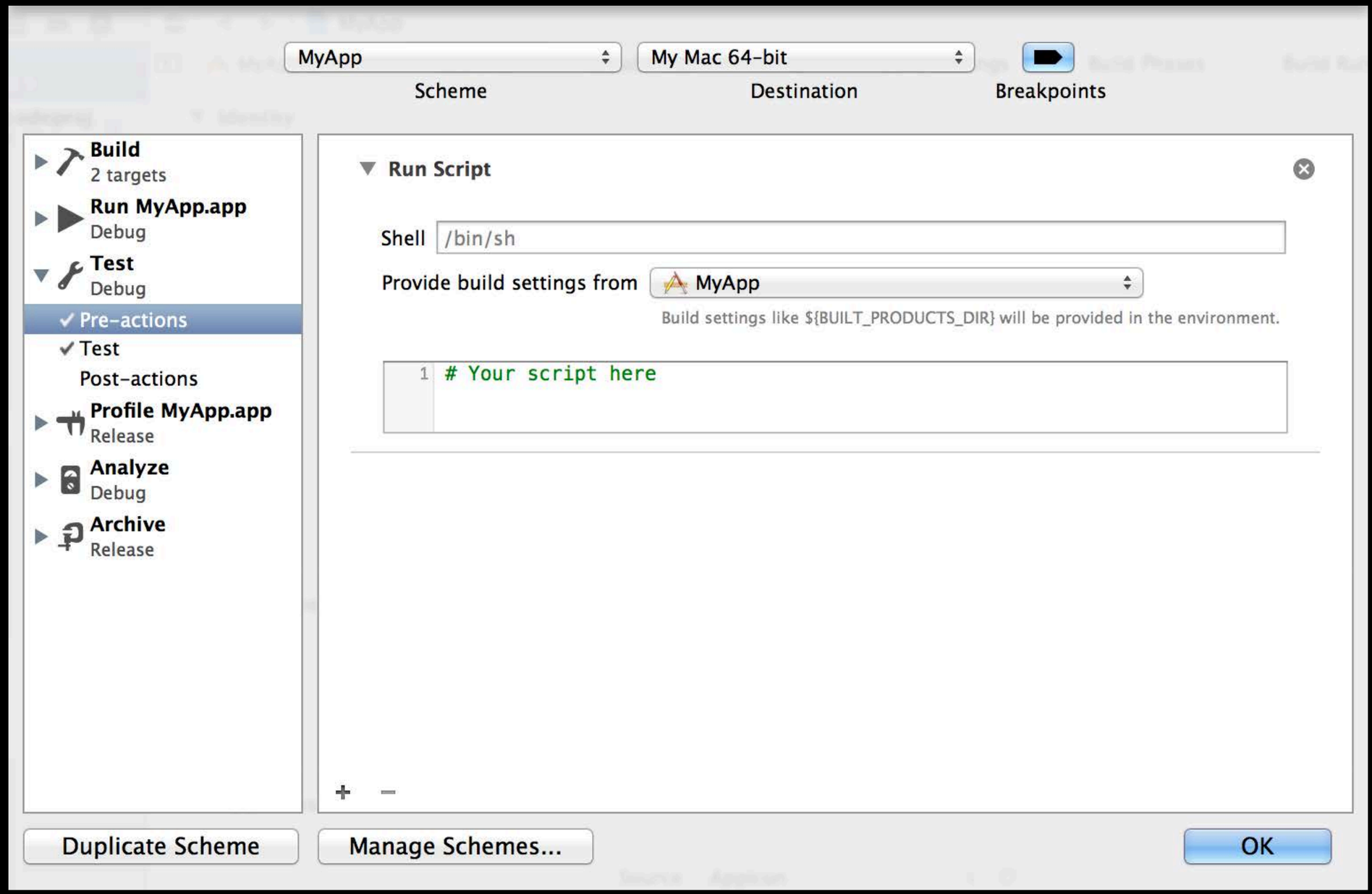
Scheme Action Build Configuration



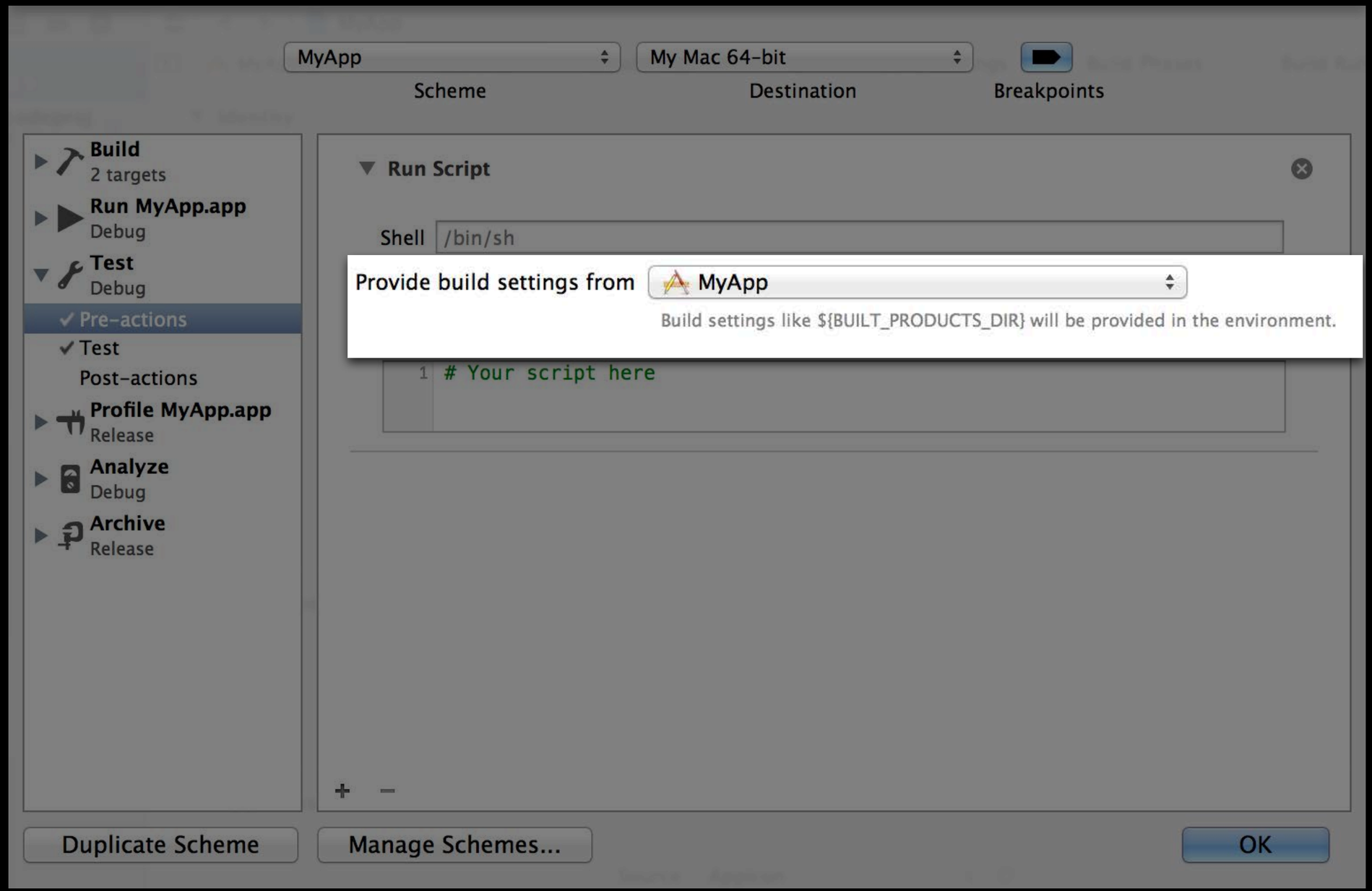
Additional Scheme Actions



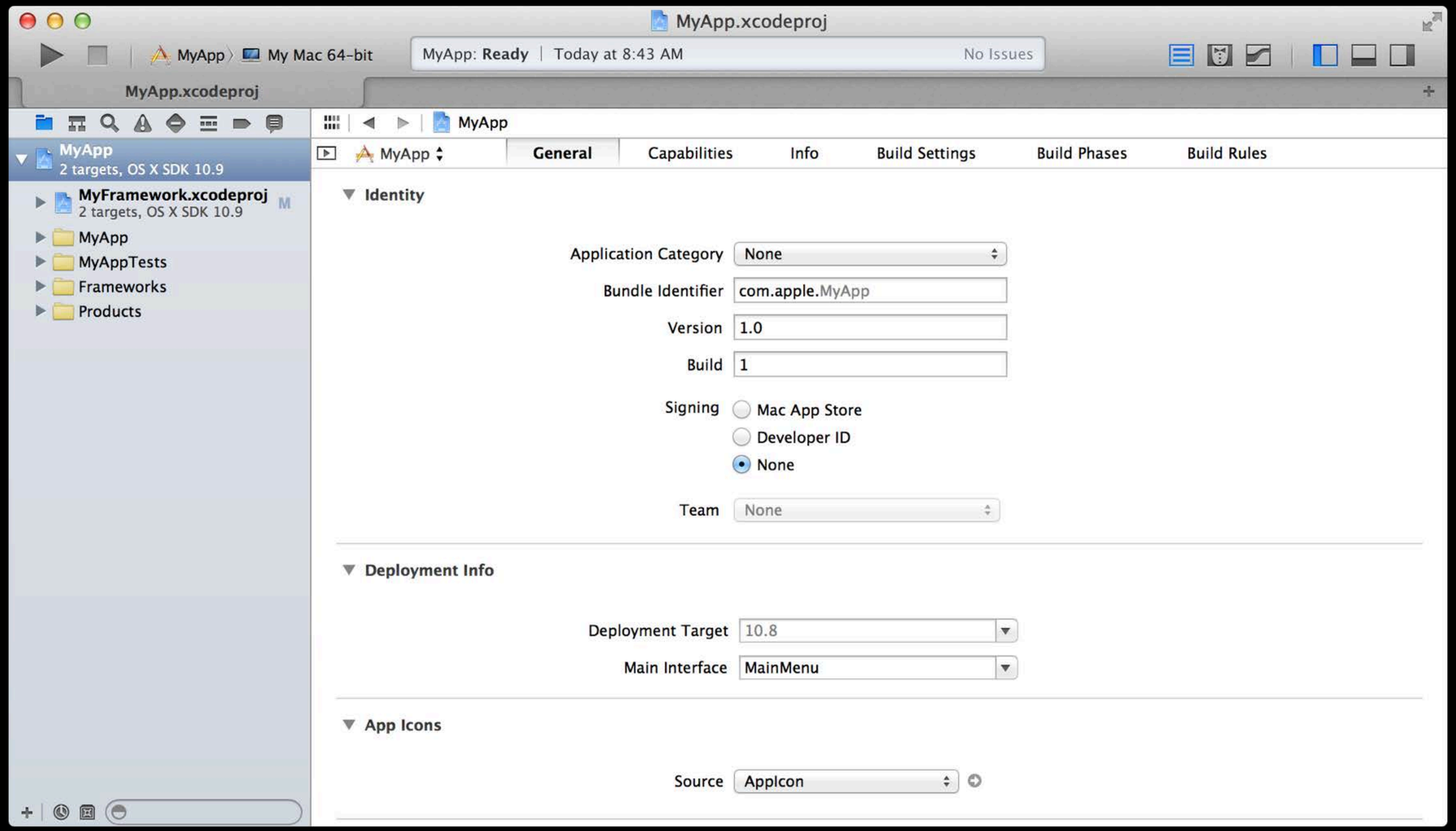
Additional Scheme Actions



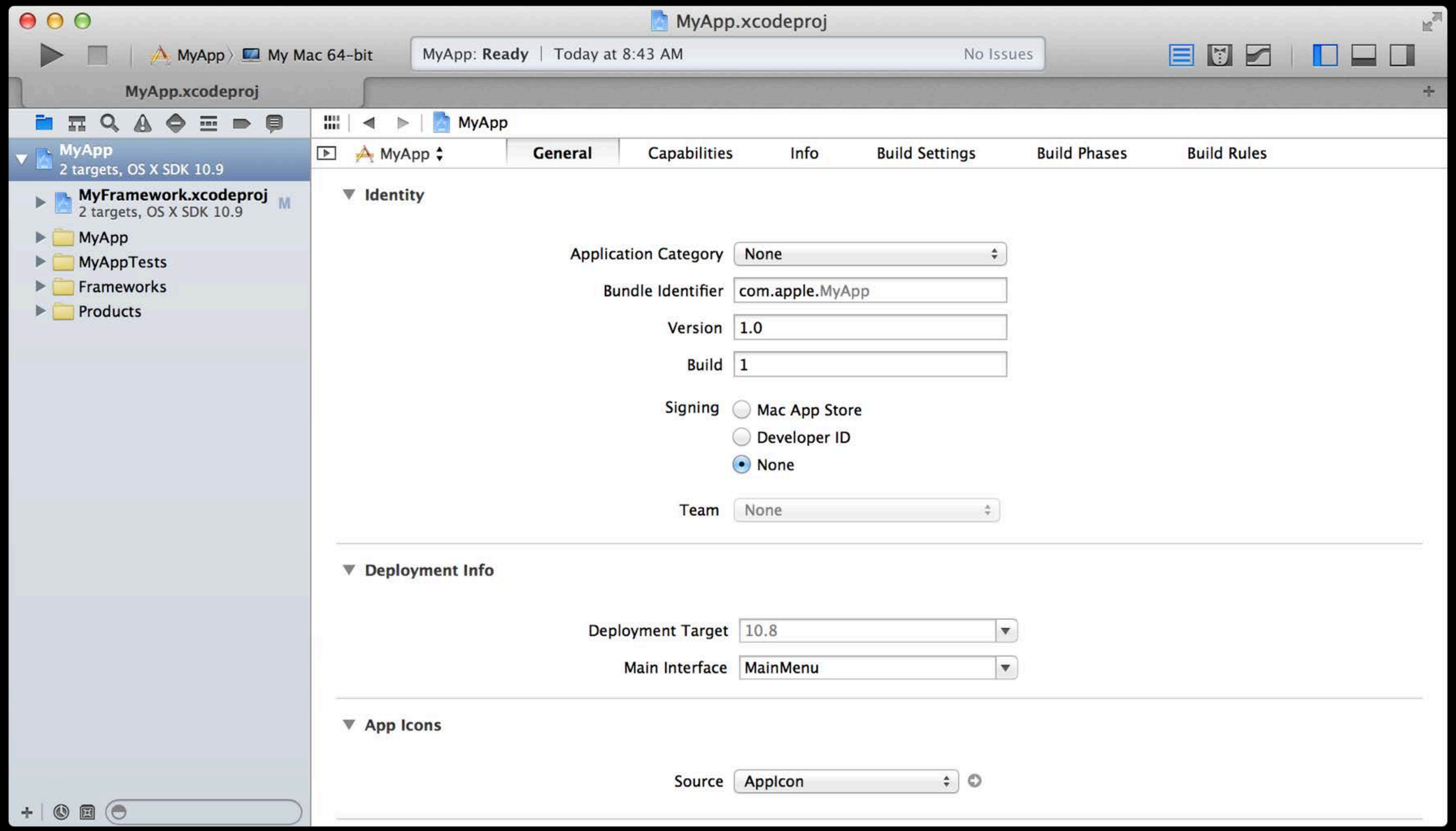
Additional Scheme Actions



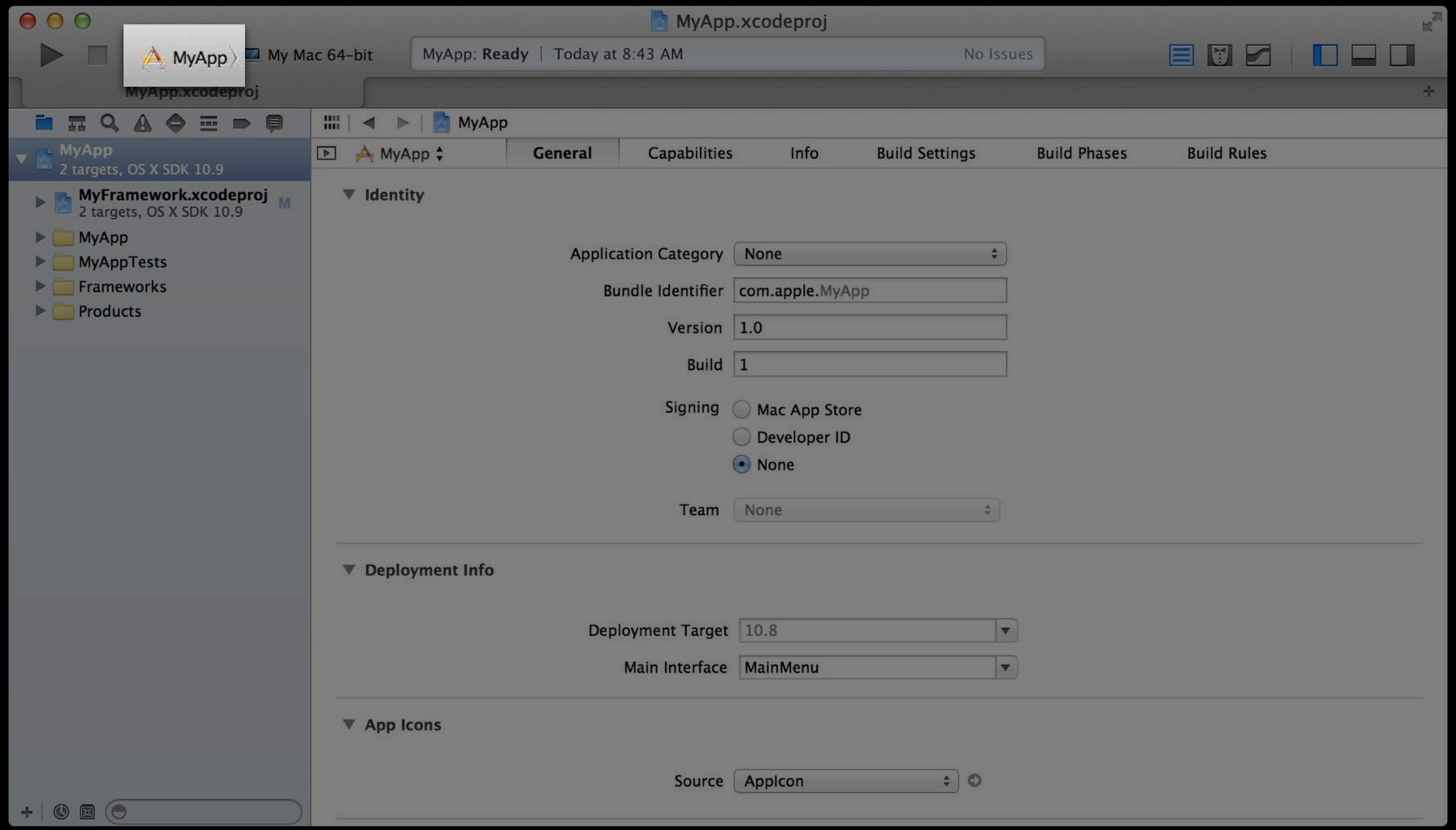
Managing Schemes



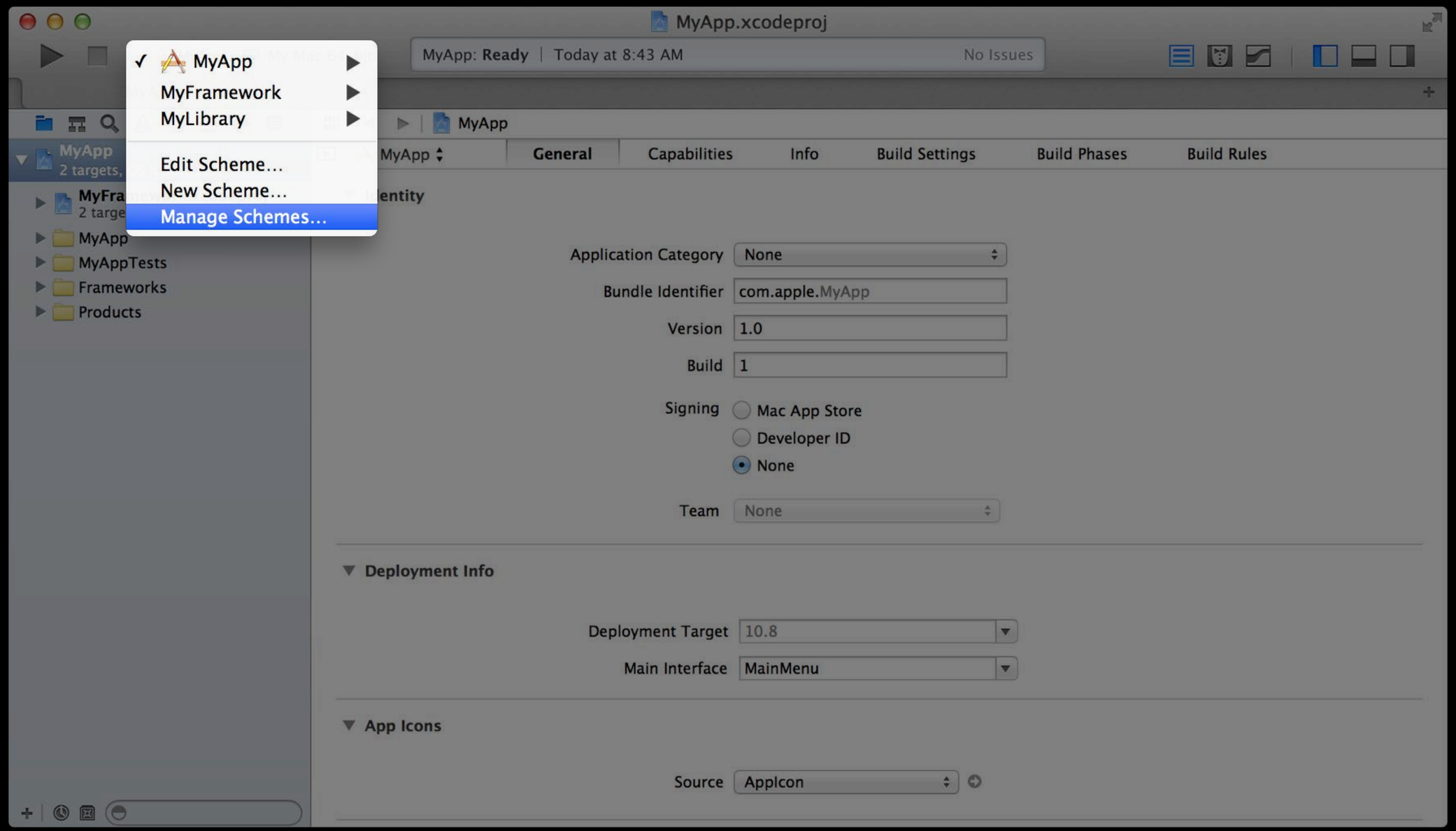
Managing Schemes



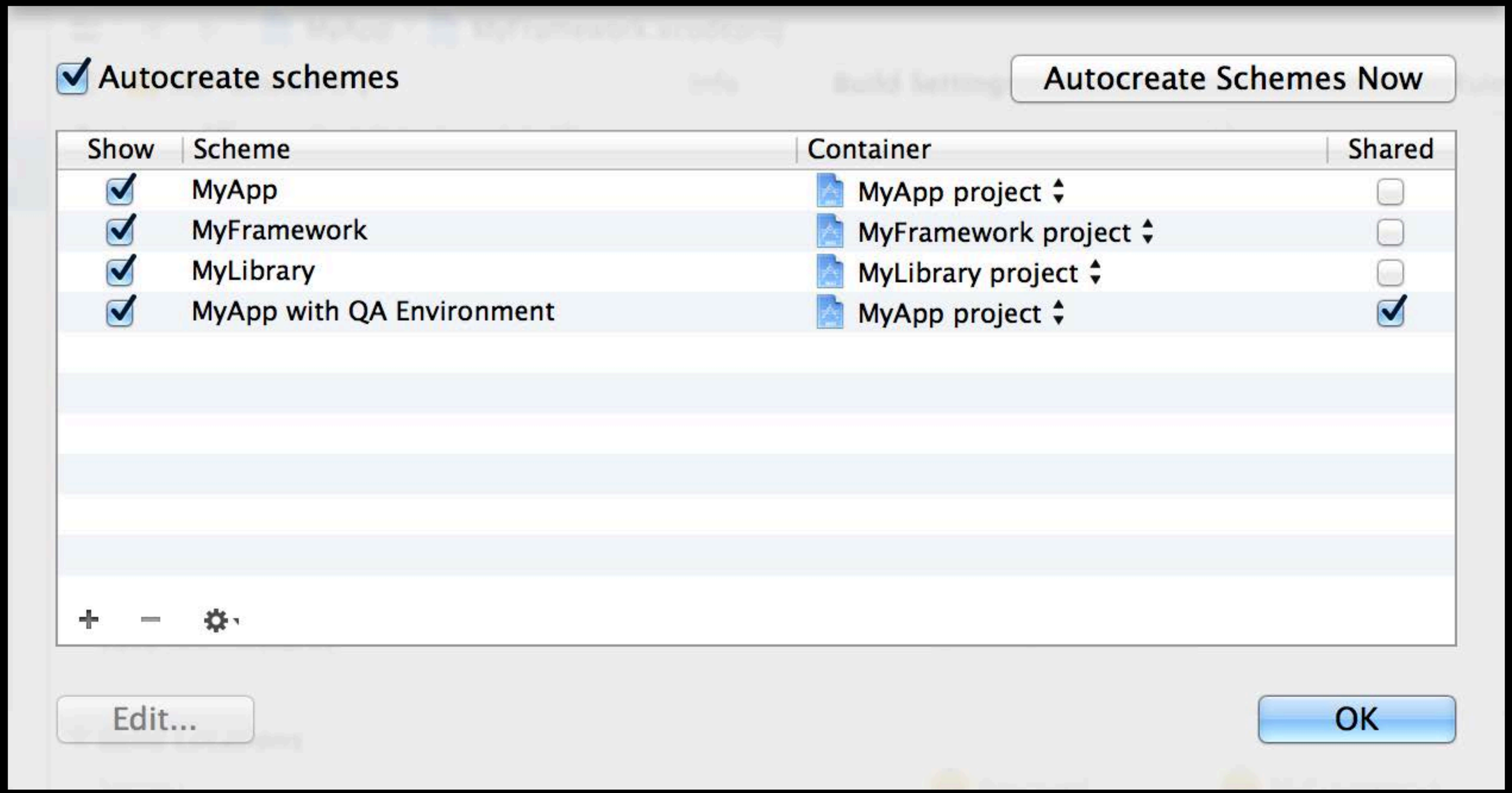
Managing Schemes



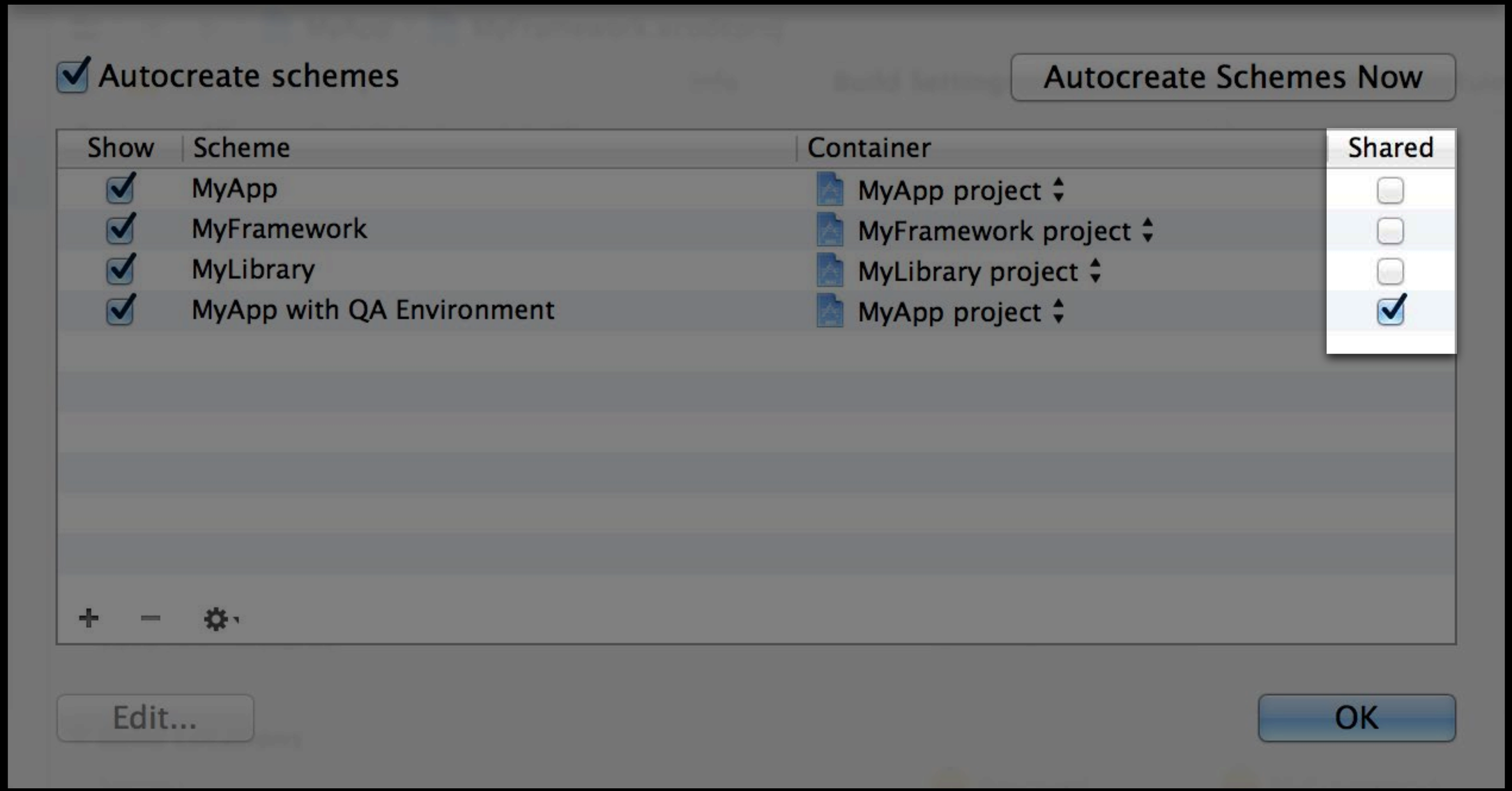
Managing Schemes



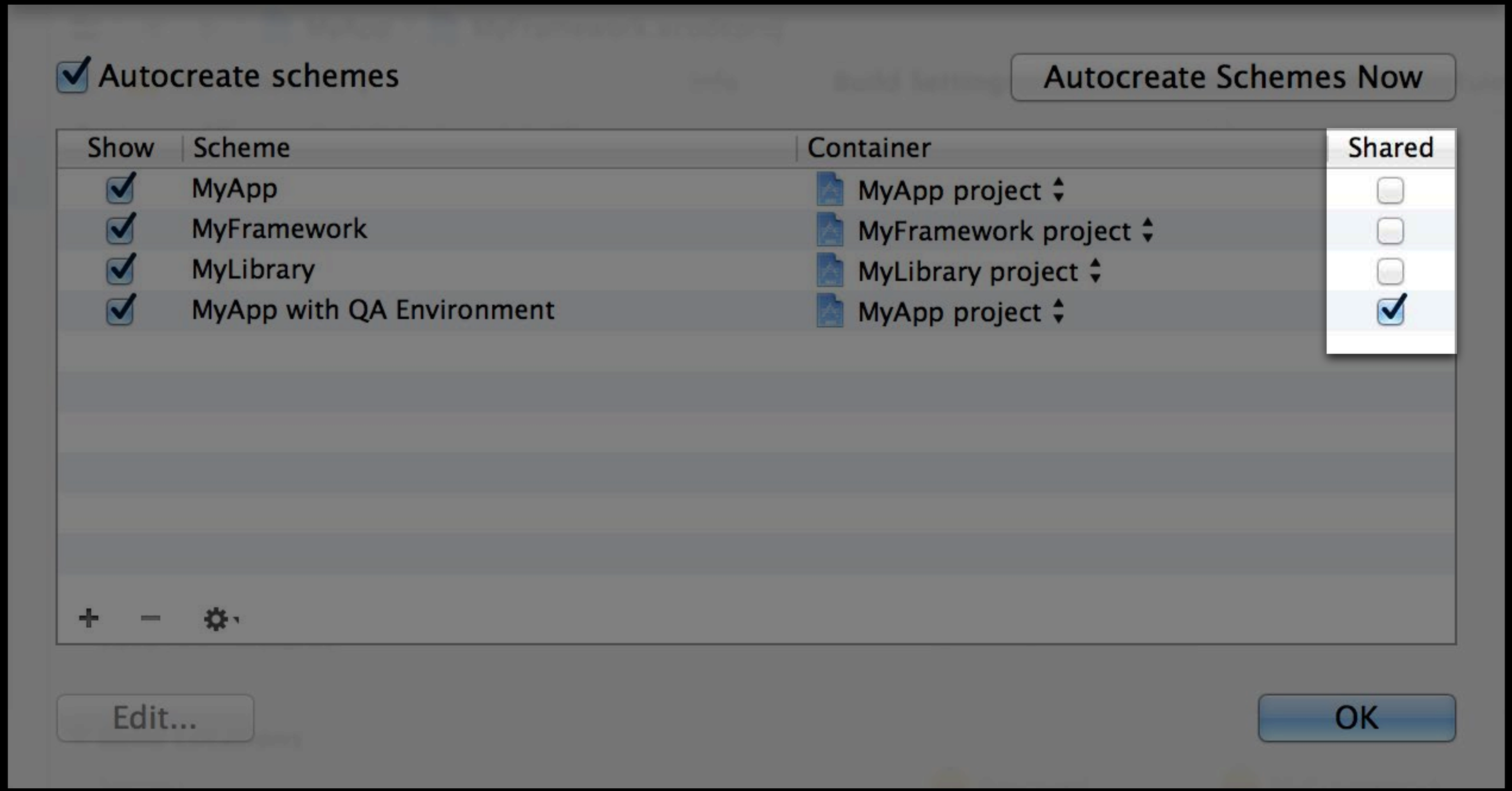
Managing Schemes



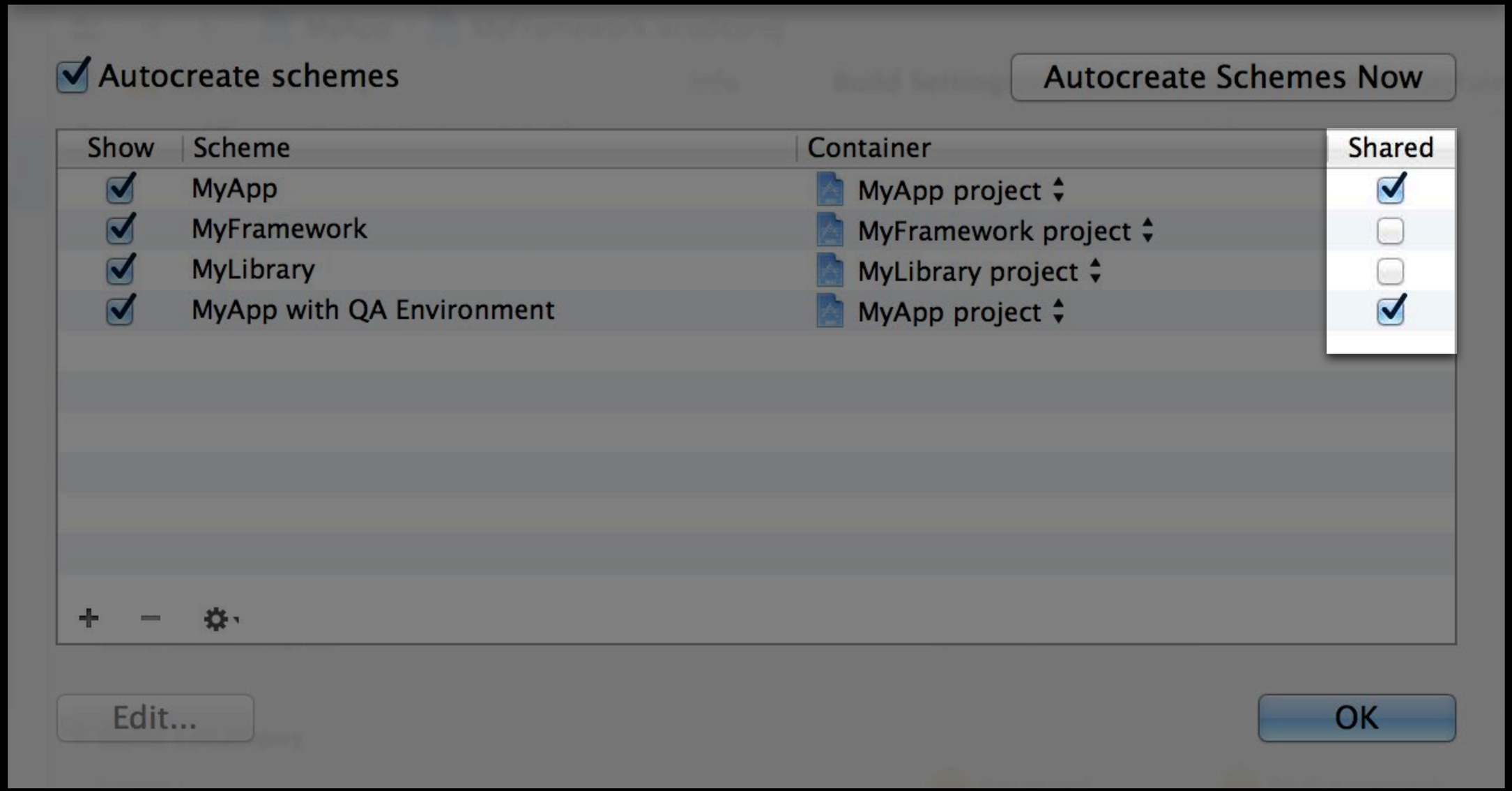
Managing Schemes



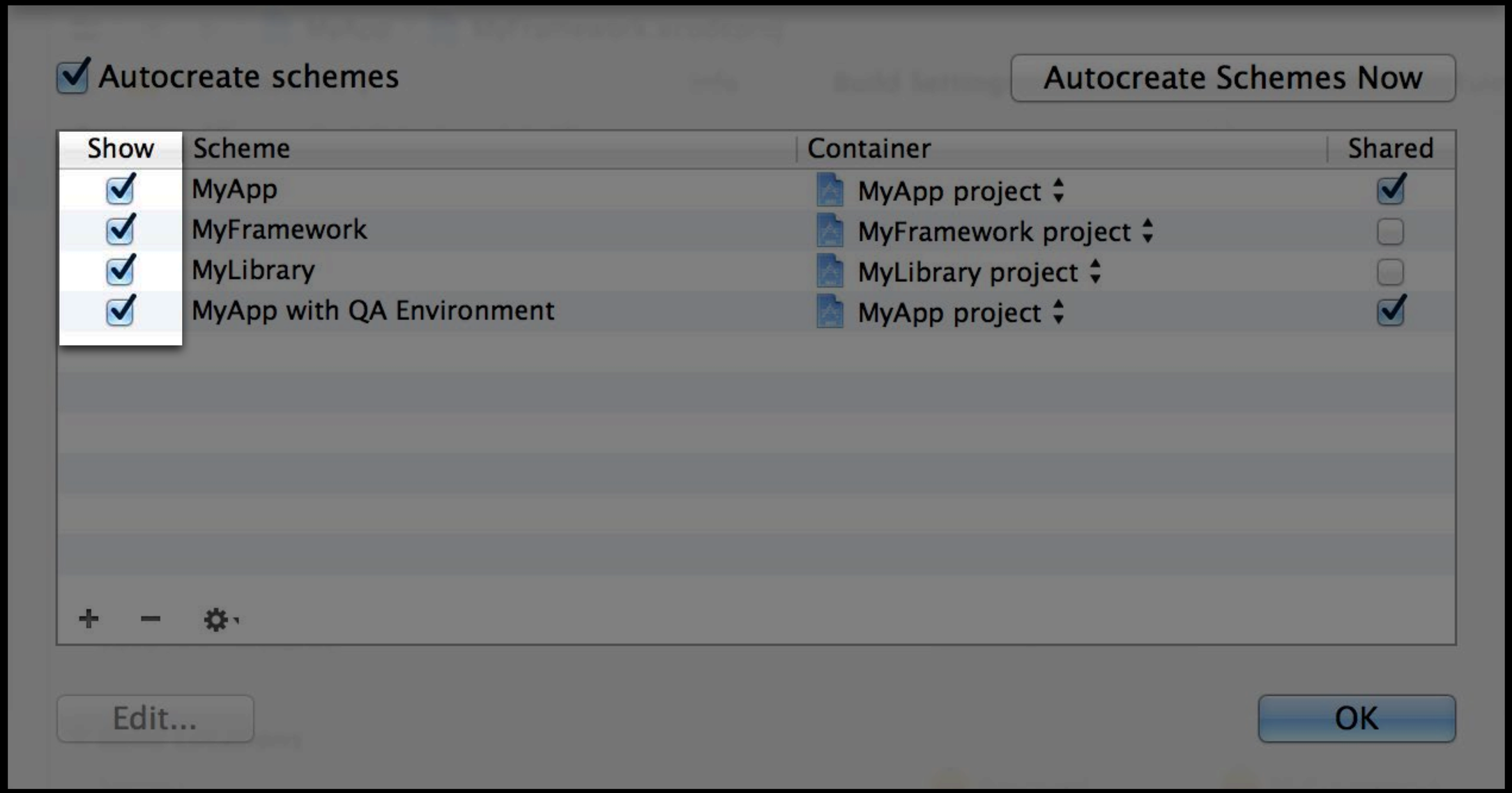
Sharing Schemes



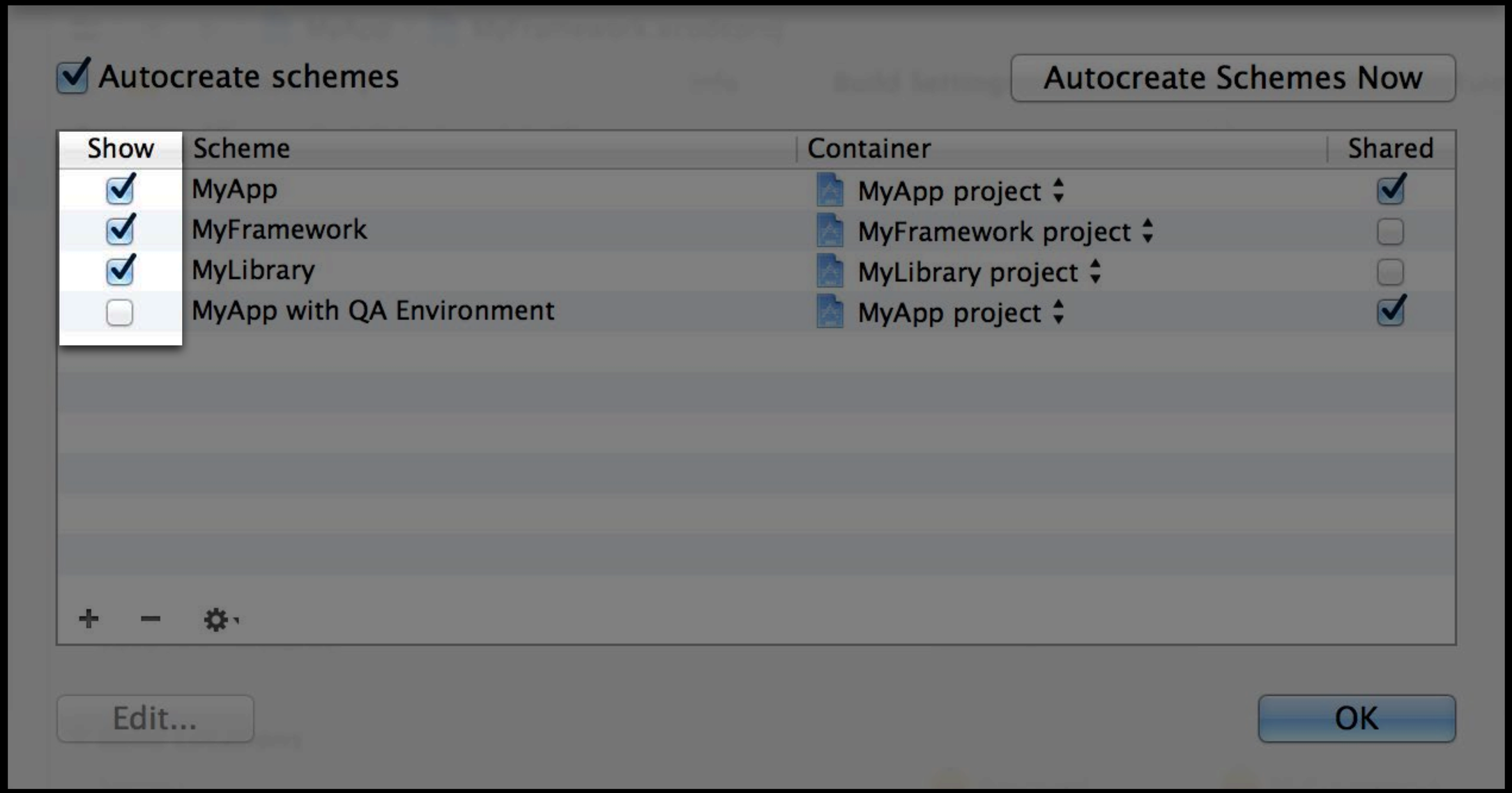
Sharing Schemes



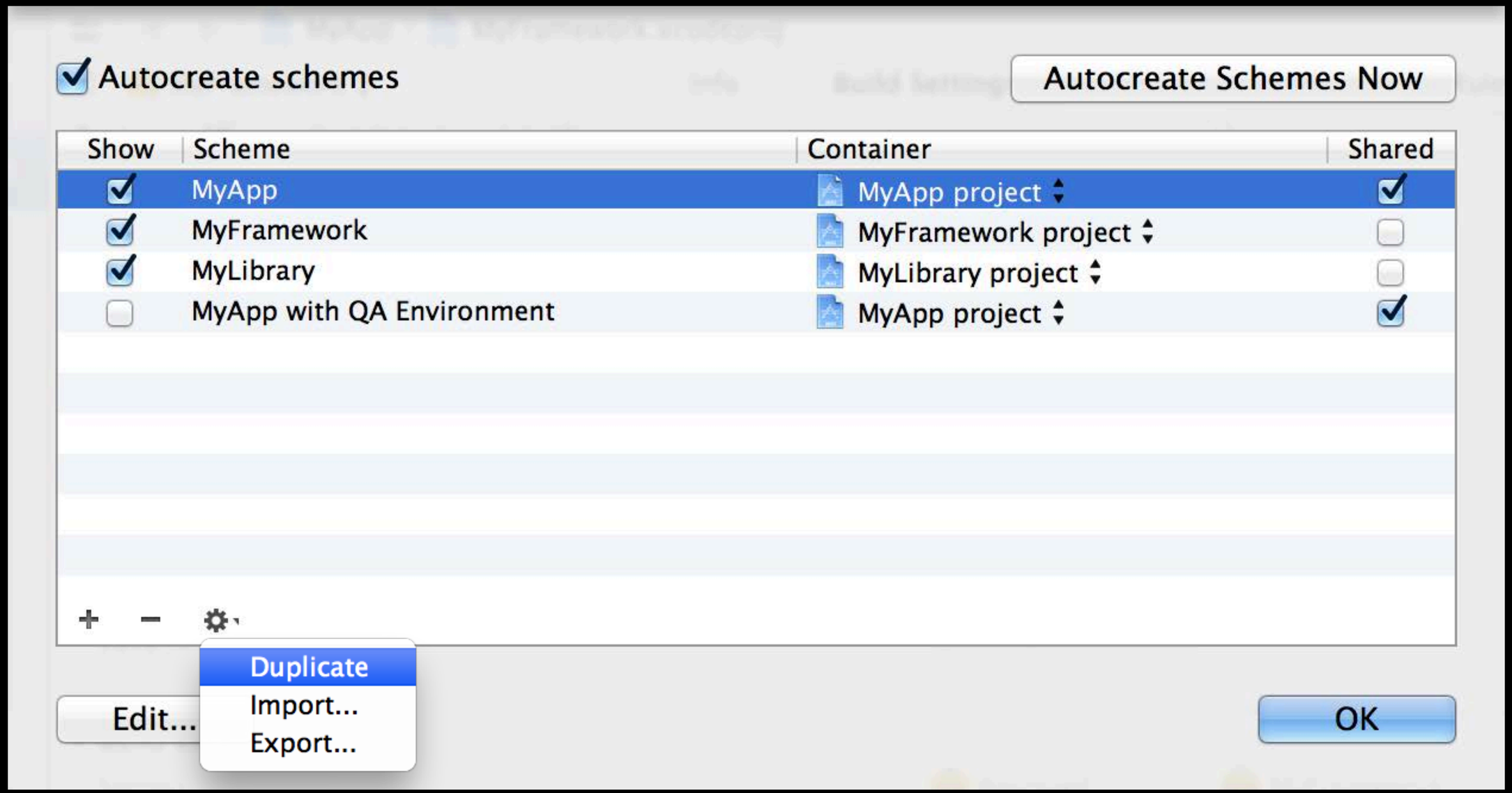
Hiding Schemes



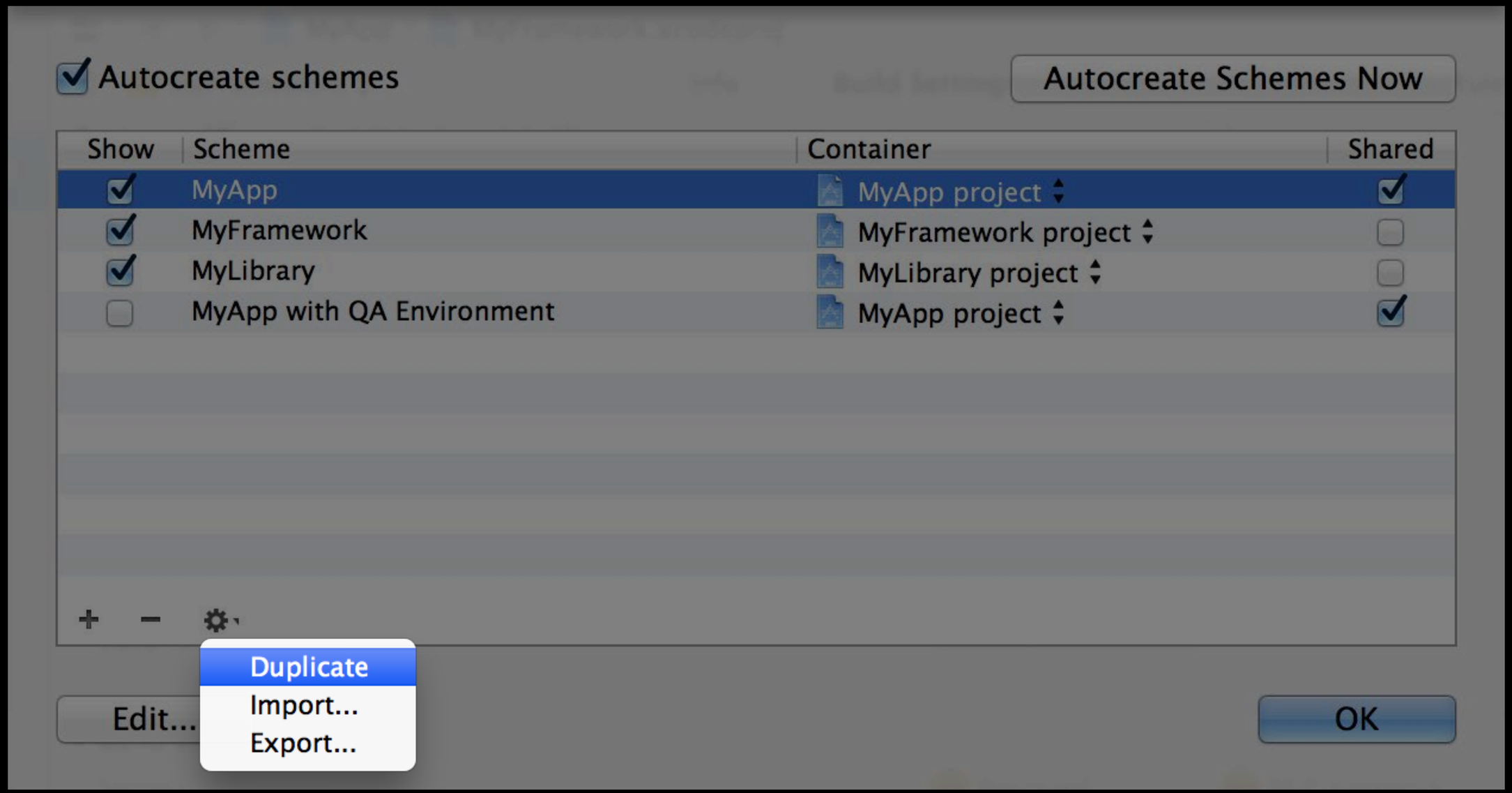
Hiding Schemes



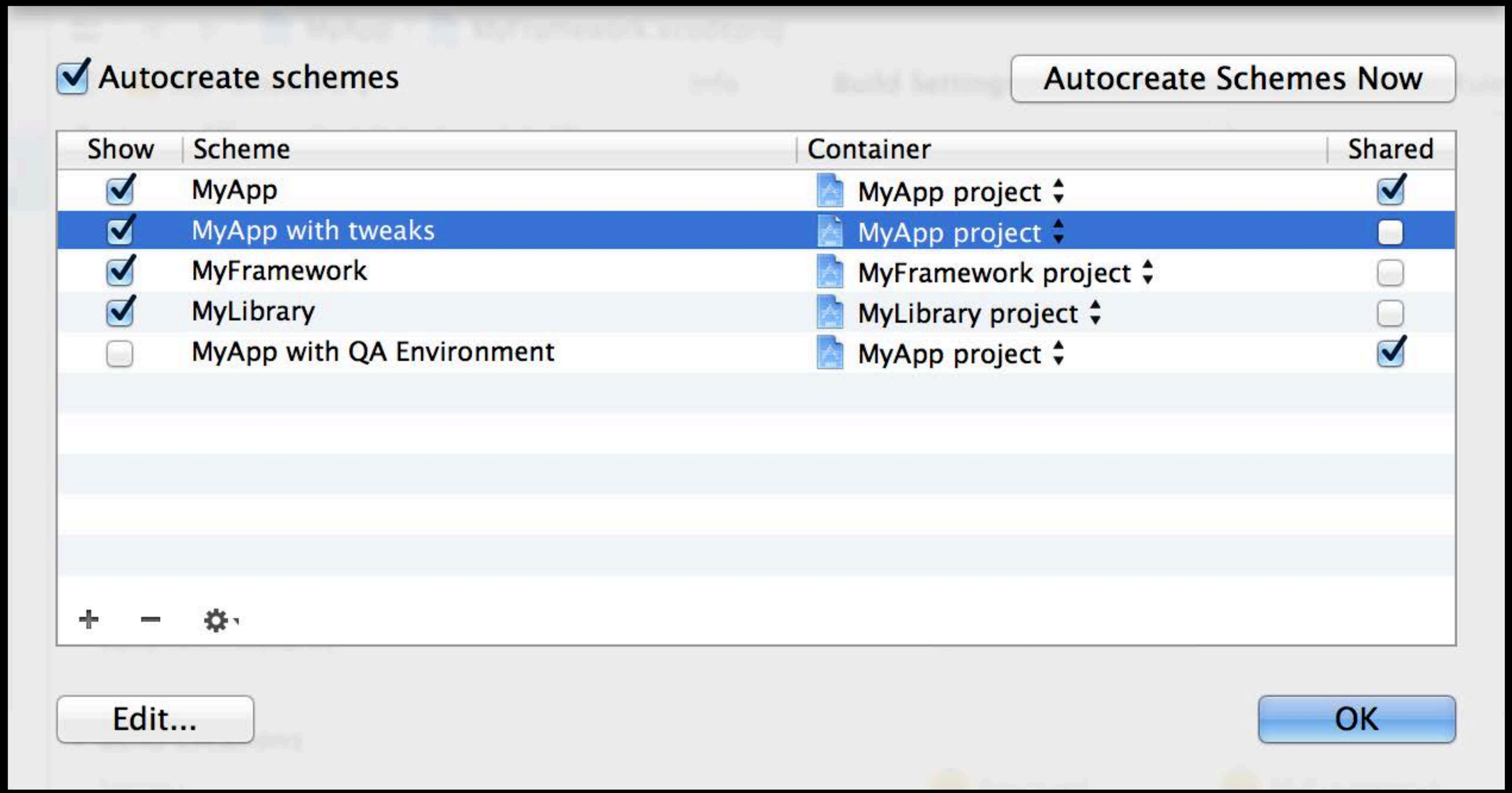
Duplicating Schemes



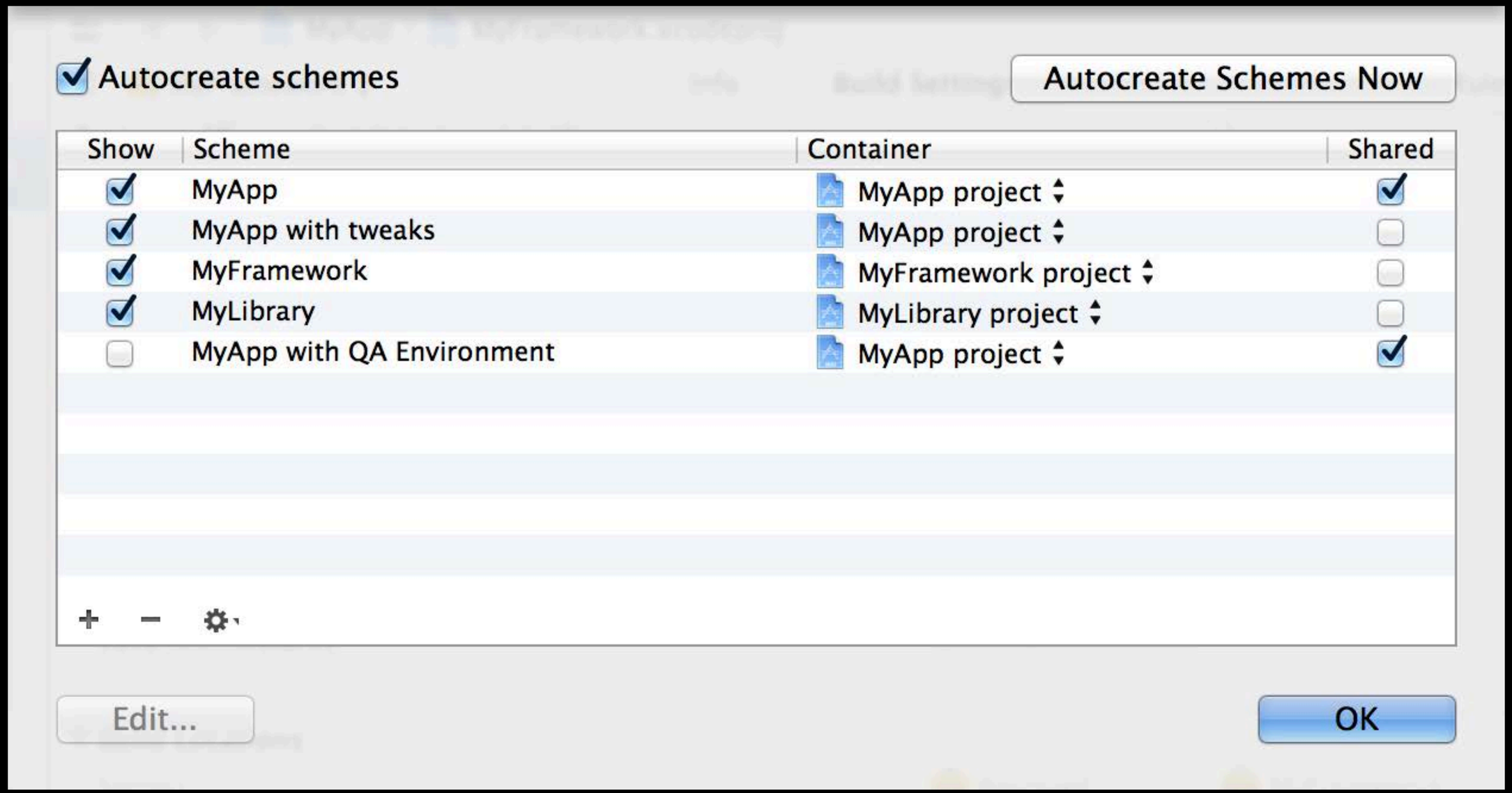
Duplicating Schemes



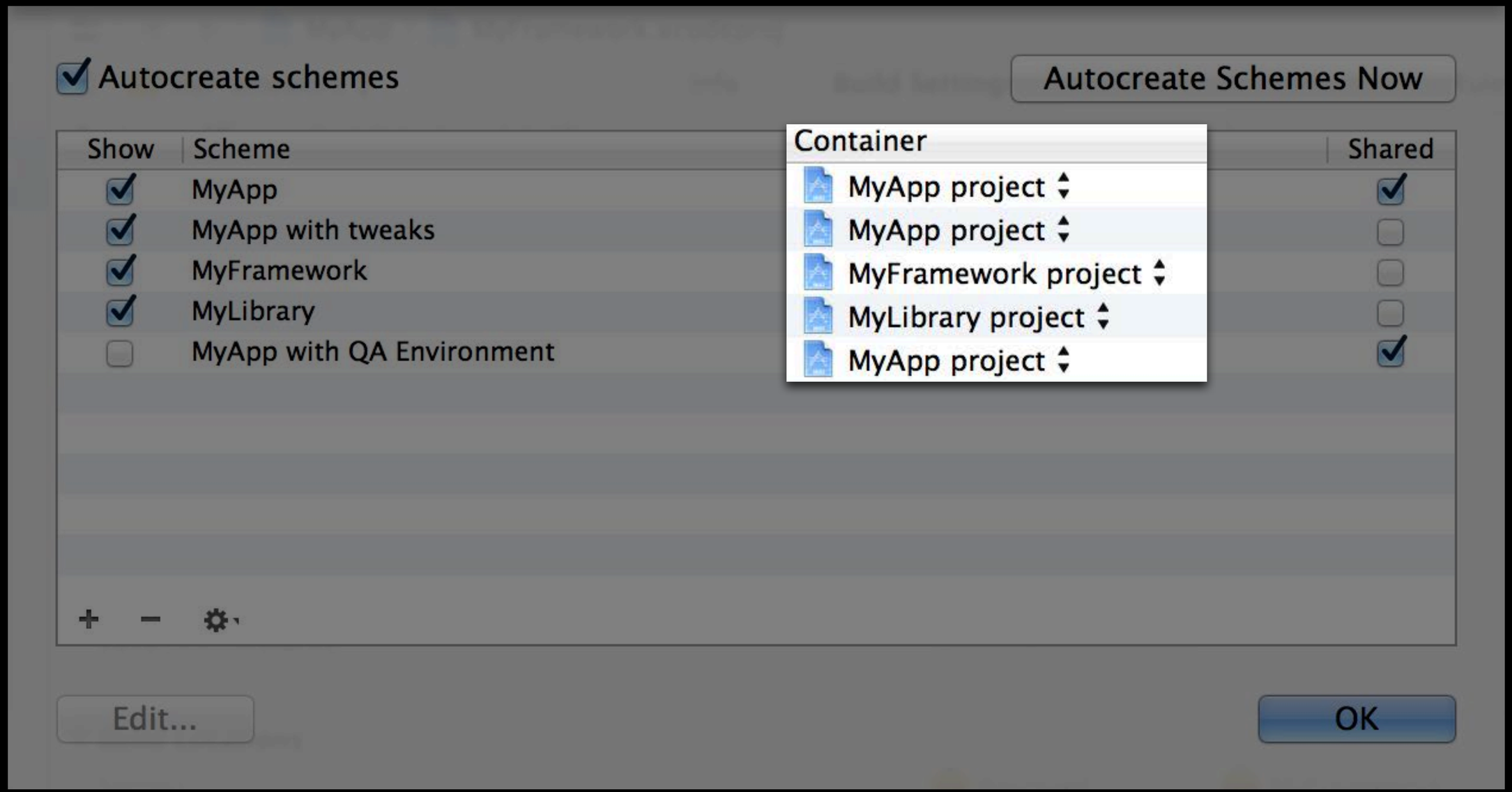
Duplicating Schemes



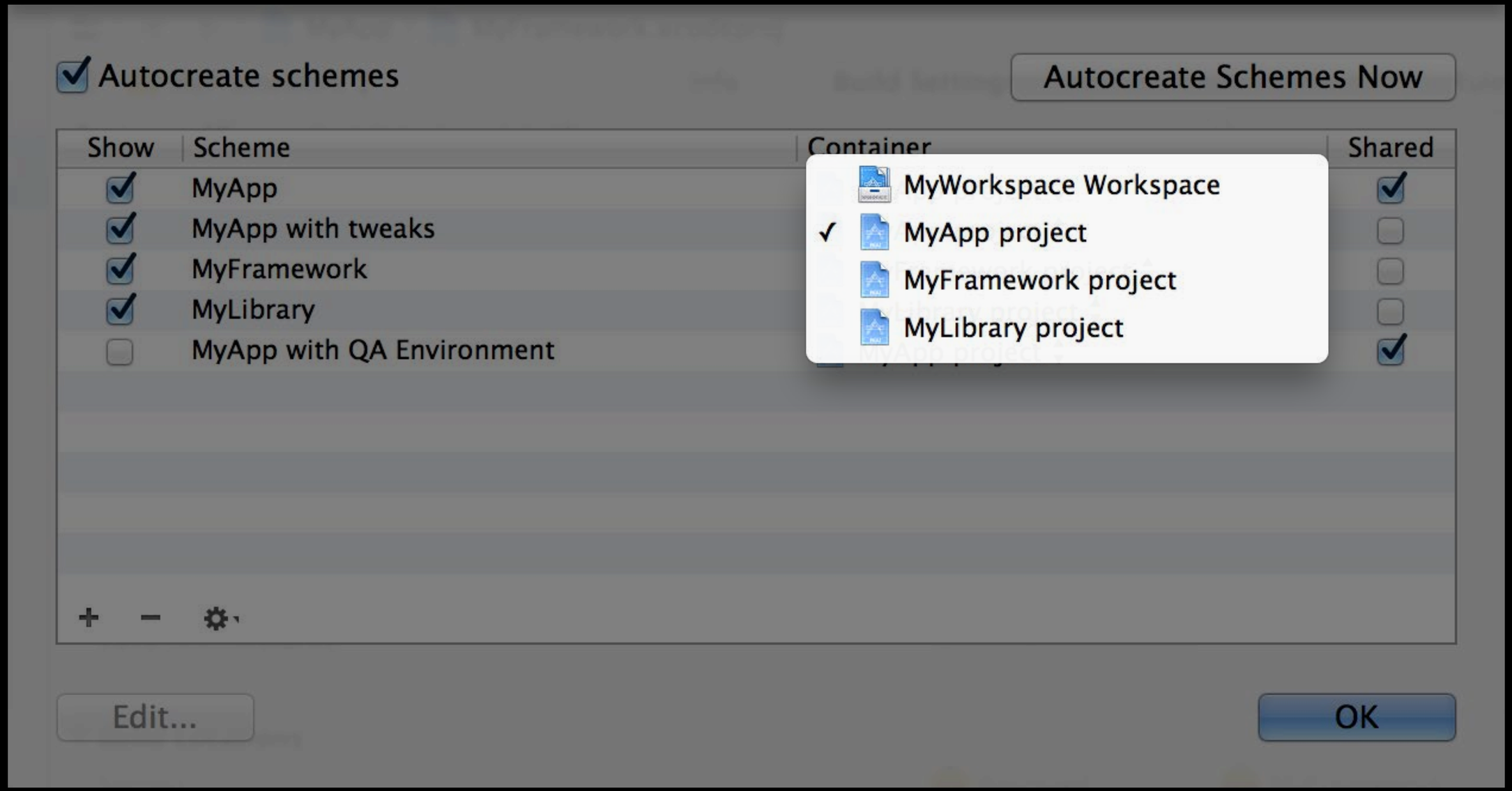
Where Schemes are Stored



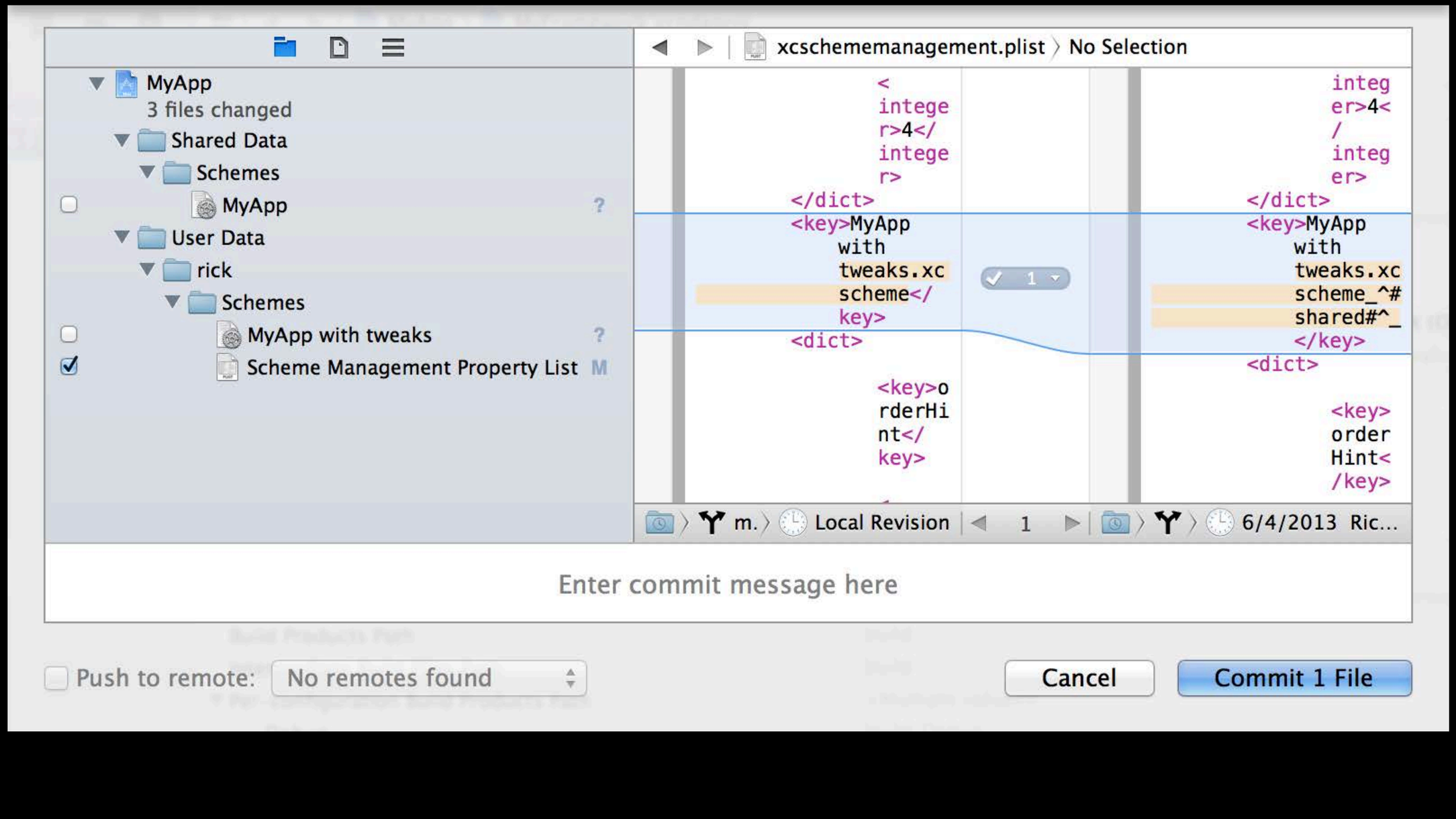
Where Schemes are Stored



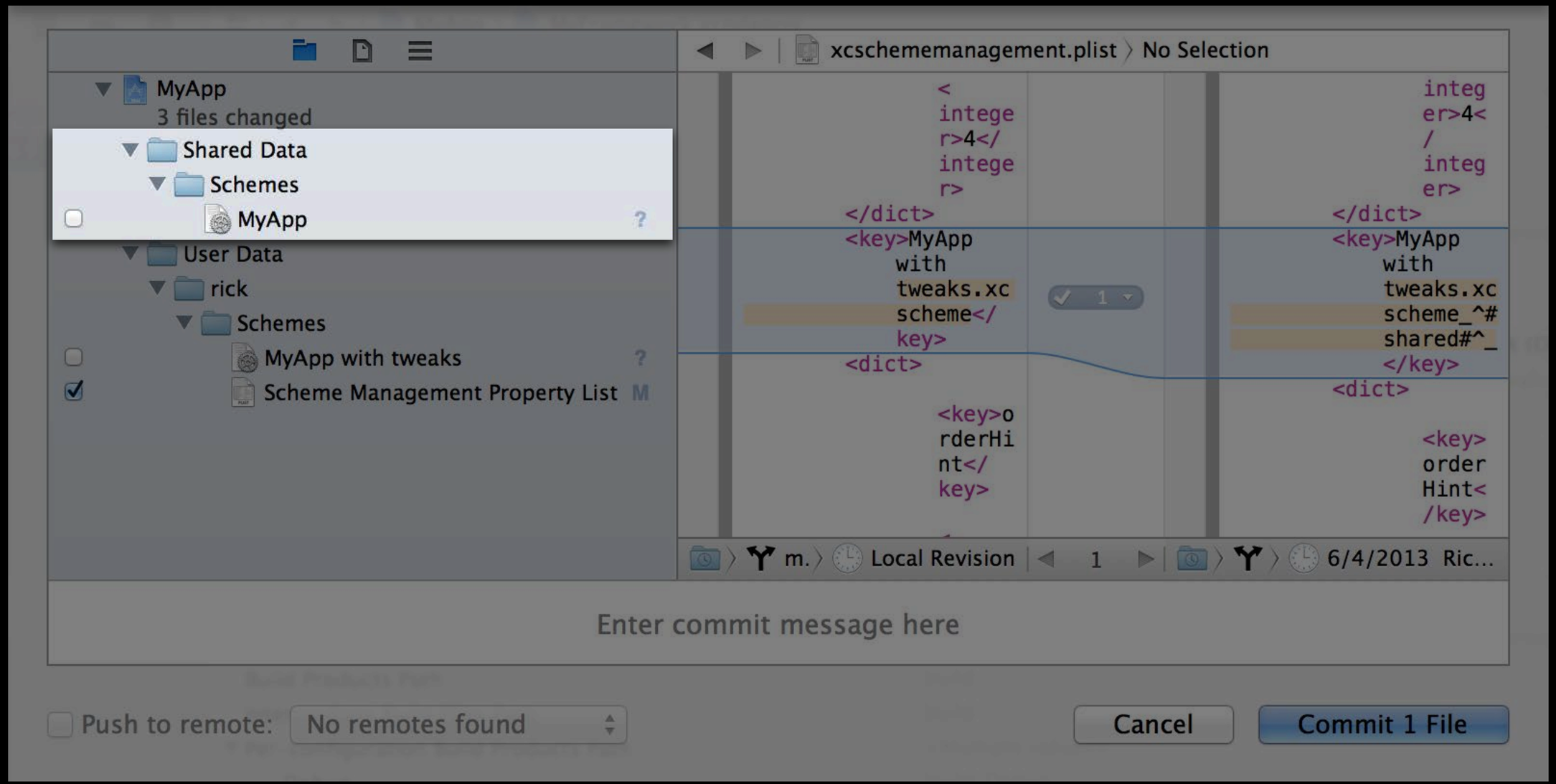
Where Schemes are Stored



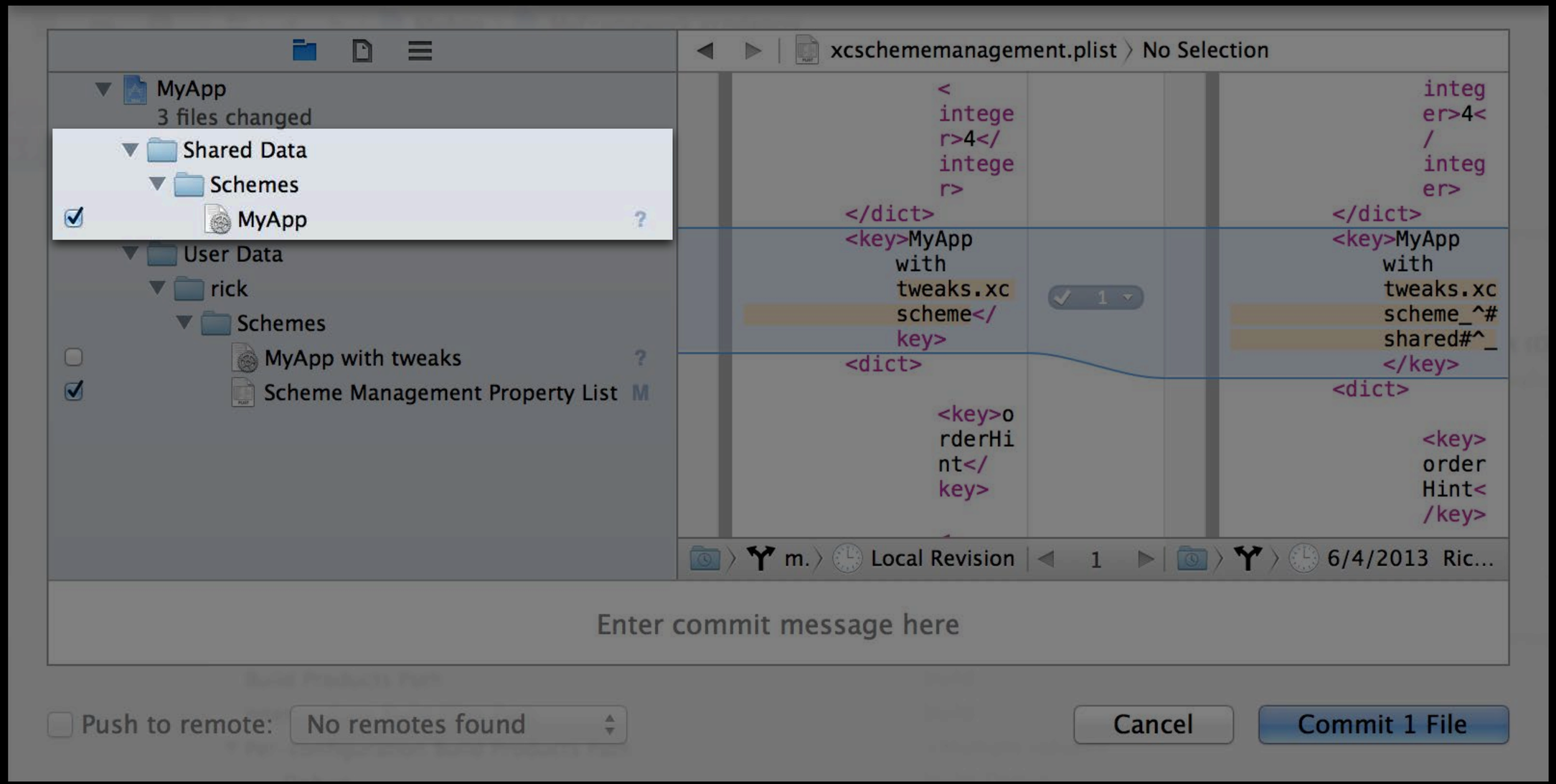
Managing Schemes in Source Control



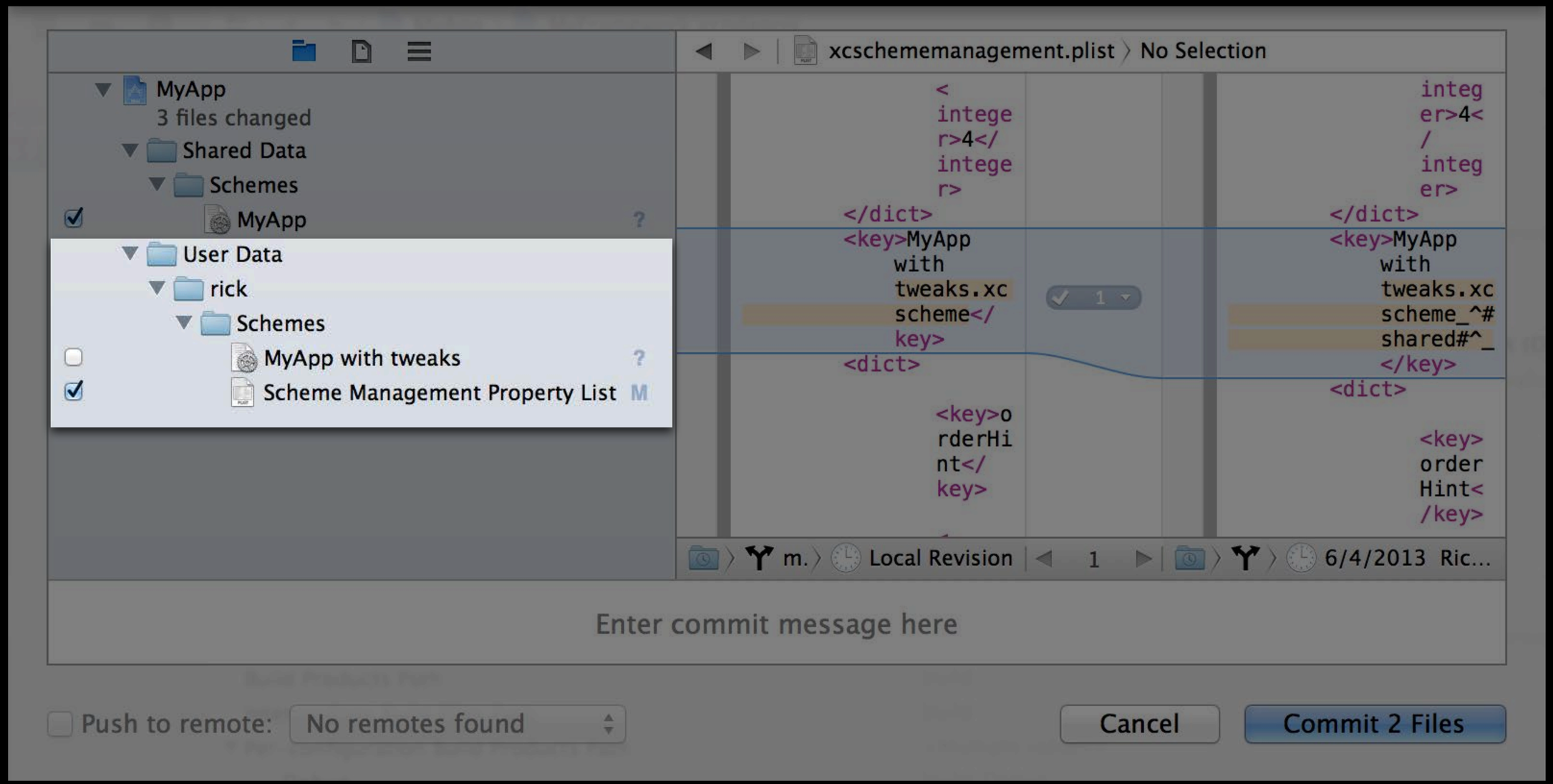
Managing Schemes in Source Control



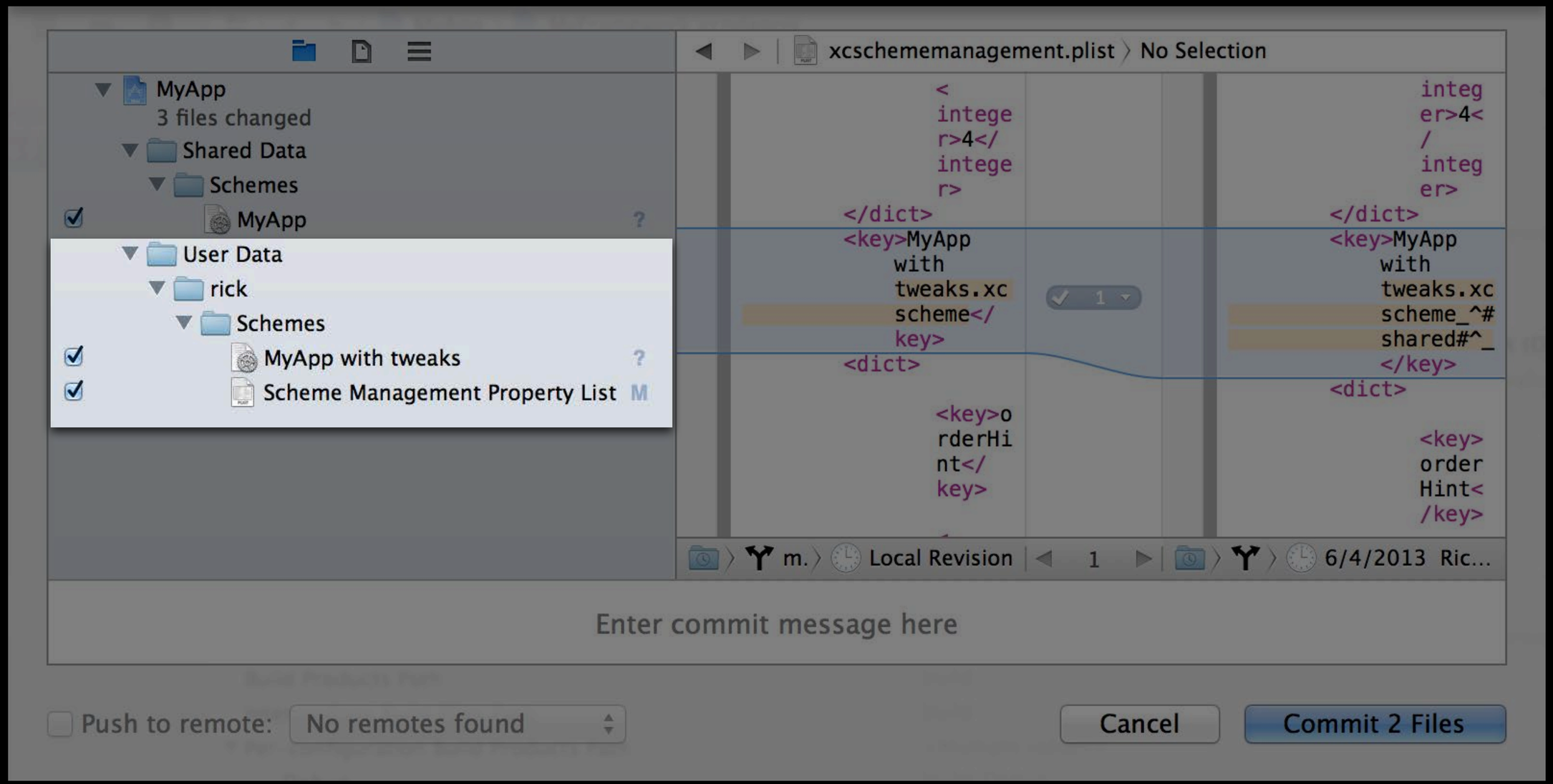
Managing Schemes in Source Control



Managing Schemes in Source Control



Managing Schemes in Source Control



Run Destinations

Devices on which you can perform scheme actions

What is a Run Destination?

- The device you want to build for and run on

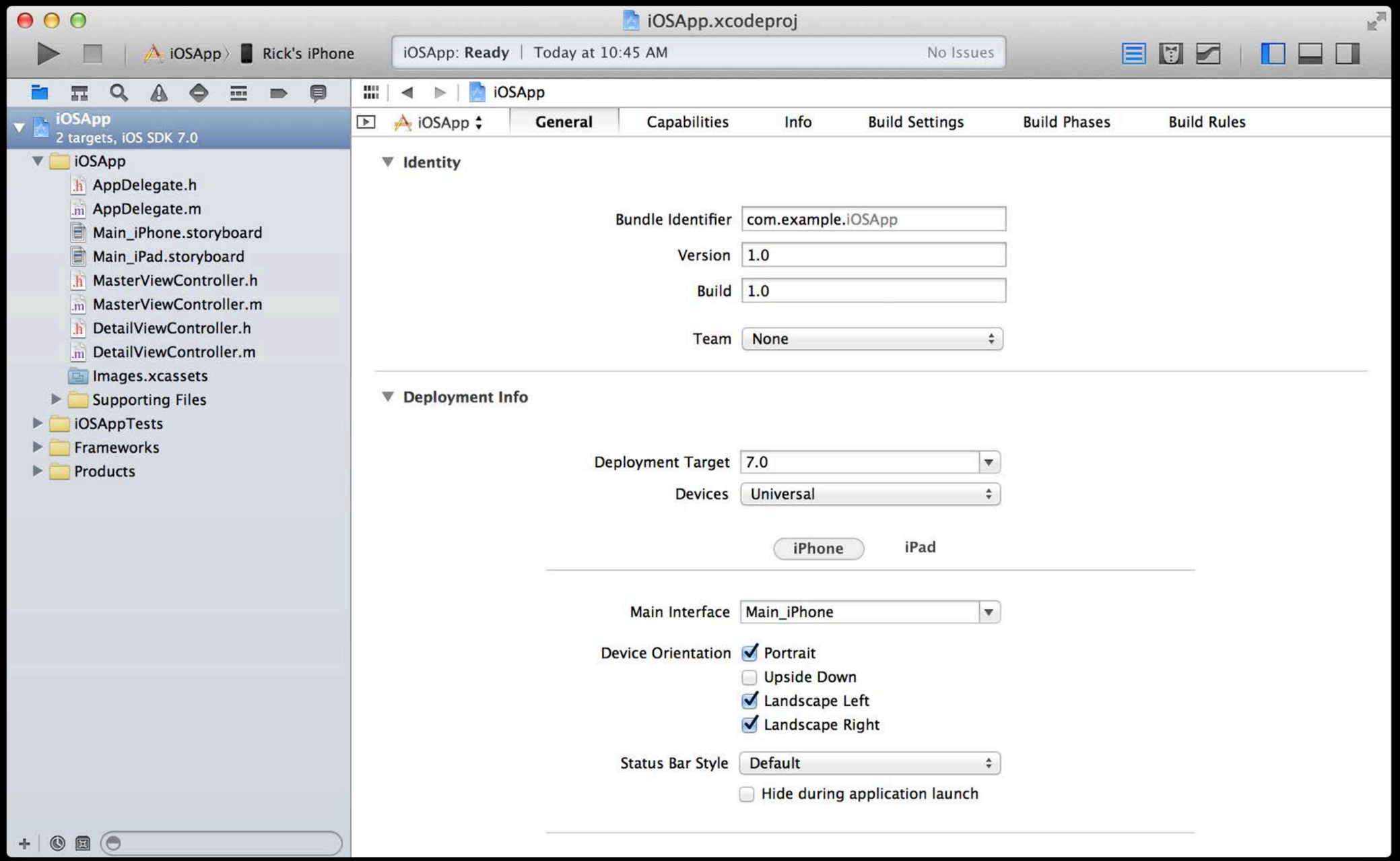
What is a Run Destination?

- The device you want to build for and run on
  - Choose among plugged-in iOS devices configured for development, available simulators, and the local Mac

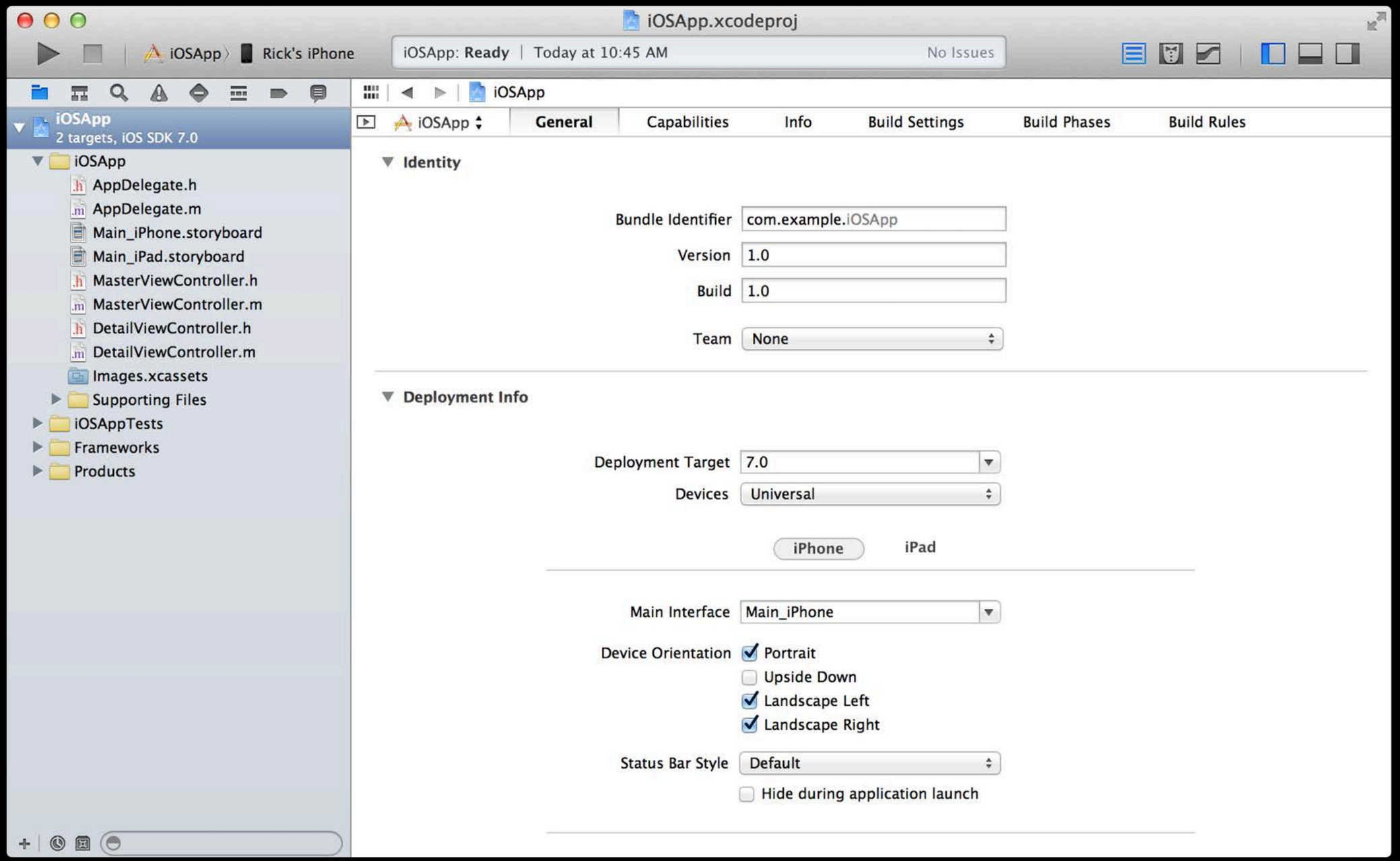
What is a Run Destination?

- The device you want to build for and run on
  - Choose among plugged-in iOS devices configured for development, available simulators, and the local Mac
  - Only devices compatible with your targets' Base SDK and Deployment Target

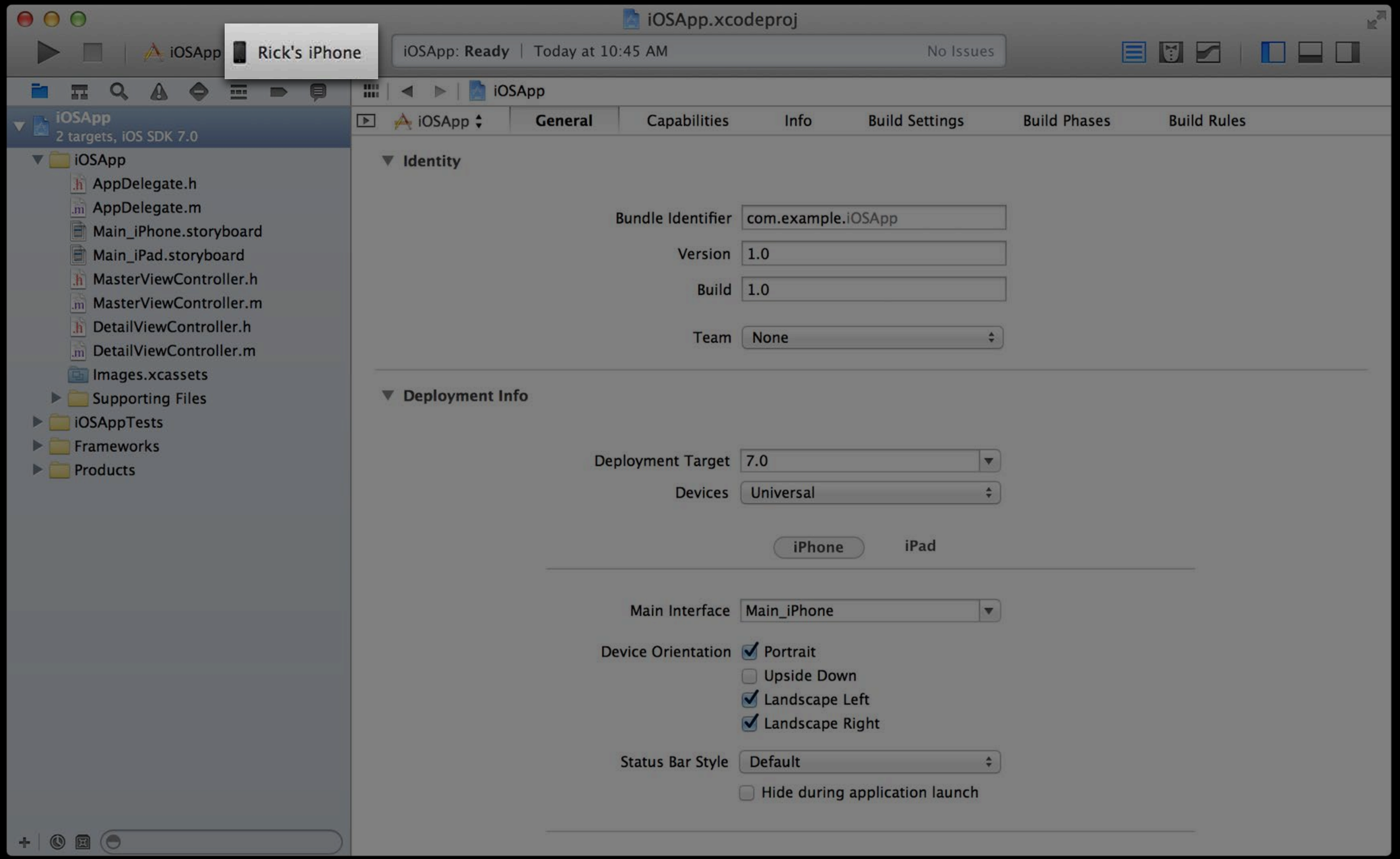
Run Destinations



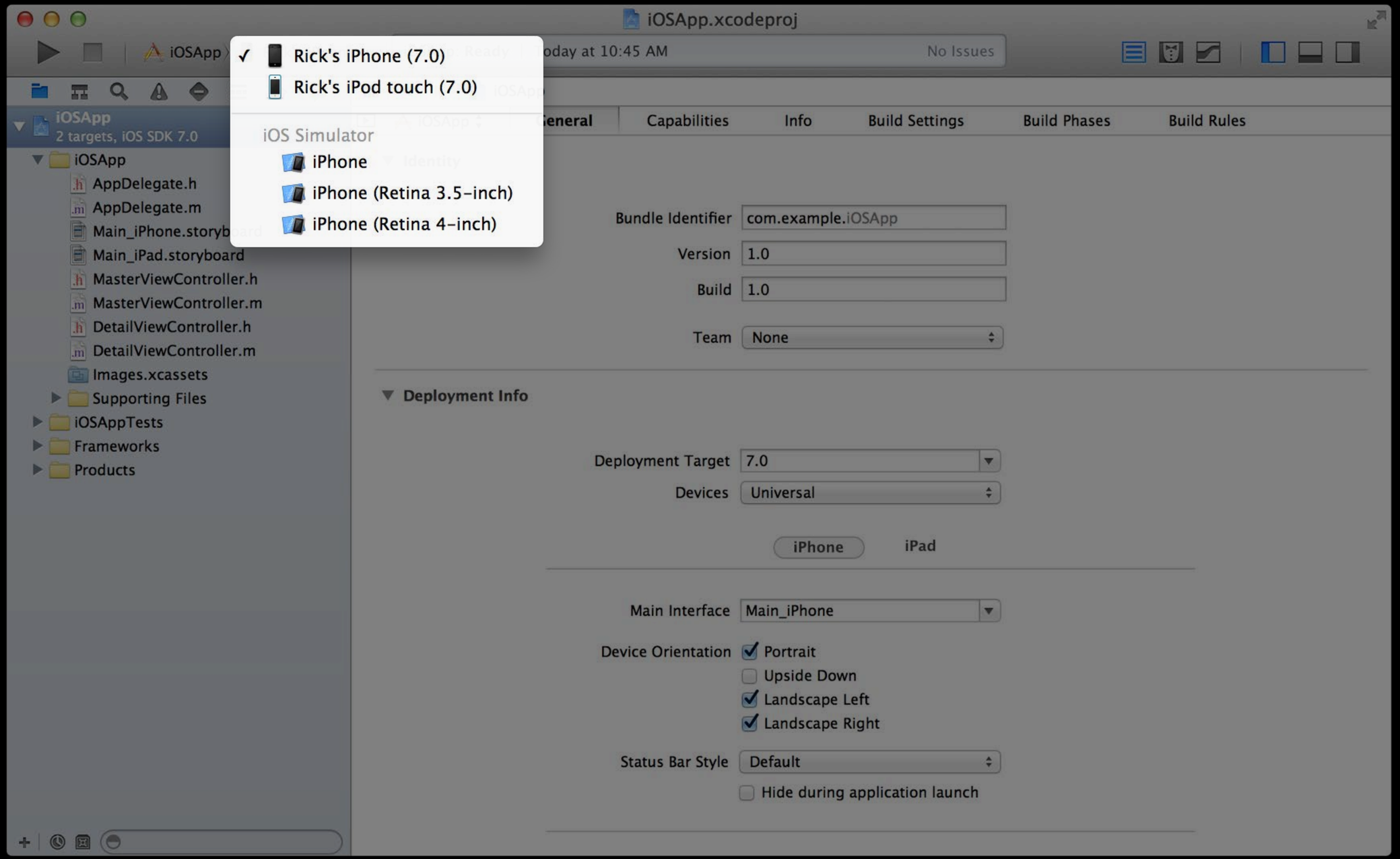
Run Destinations



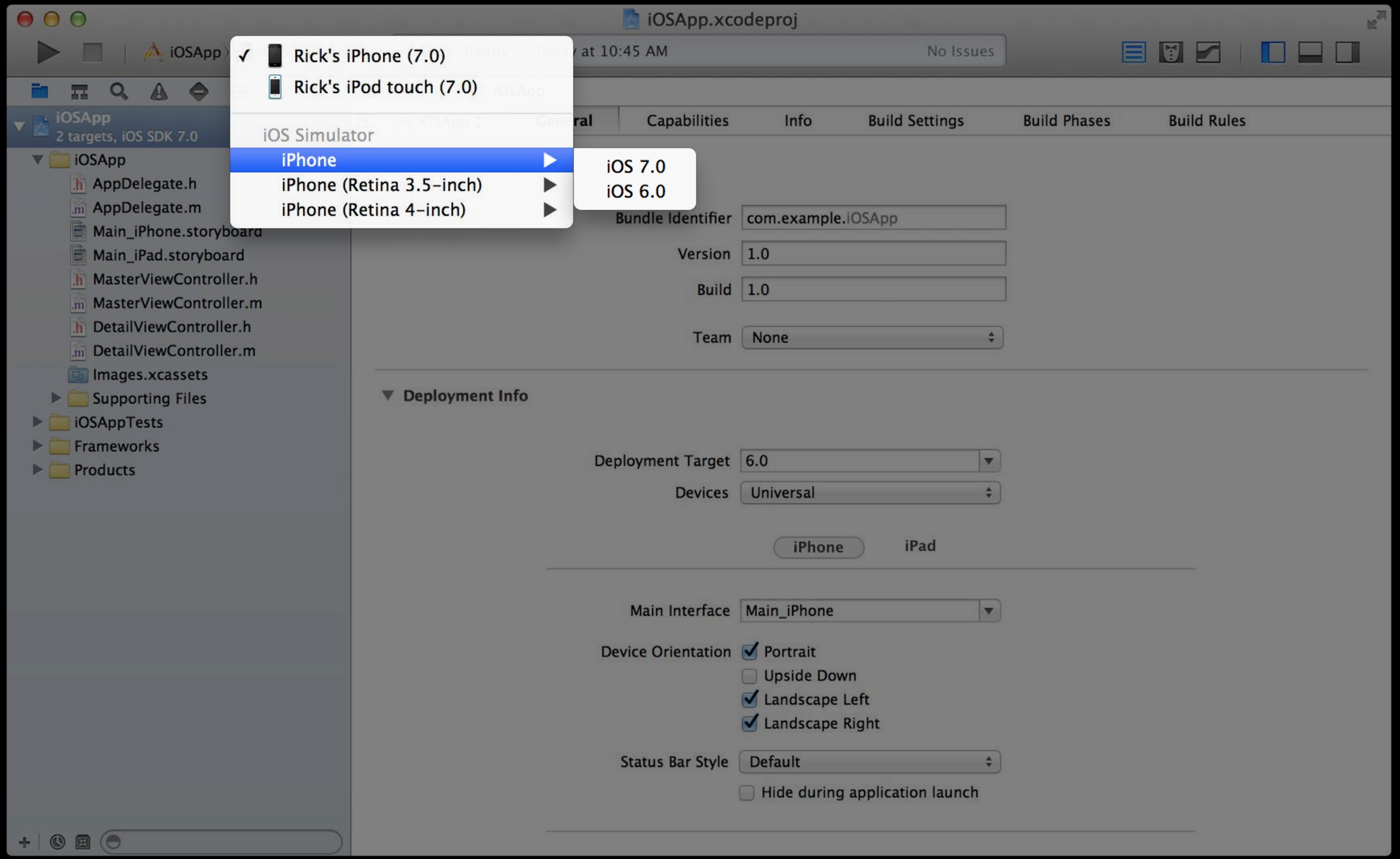
Run Destinations



Run Destinations



Run Destinations



Fundamental Concepts

Fundamental Concepts

- Workspaces

Fundamental Concepts

- Workspaces
- Projects

Fundamental Concepts

- Workspaces
- Projects
- Targets

Fundamental Concepts

- Workspaces
- Projects
- Targets
- Schemes

Fundamental Concepts

- Workspaces
- Projects
- Targets
- Schemes
- Run destinations

Takeaways

Takeaways

- Navigate and edit efficiently in Xcode

Takeaways

- Navigate and edit efficiently in Xcode
- Integrate your documentation into Xcode

Takeaways

- Navigate and edit efficiently in Xcode
- Integrate your documentation into Xcode
- Customize Xcode’s behaviors

Takeaways

- Navigate and edit efficiently in Xcode
- Integrate your documentation into Xcode
- Customize Xcode's behaviors
- Create and configure workspaces, projects, and targets

Takeaways

- Navigate and edit efficiently in Xcode
- Integrate your documentation into Xcode
- Customize Xcode's behaviors
- Create and configure workspaces, projects, and targets
- Manage and share your schemes

More Information

Dave DeLong
App Frameworks and Developer Tools Evangelist
delong@apple.com

Documentation
Xcode Guidelines
http://developer.apple.com/xcode/

Apple Developer Forums
http://devforums.apple.com

Related Sessions

Continuous Integration with Xcode 5	Presidio Tuesday 3:15PM
Debugging with Xcode	Pacific Heights Wednesday 2:00PM
Testing in Xcode 5	Pacific Heights Thursday 9:00AM
Understanding Source Control in Xcode	Pacific Heights Friday 10:15AM
Interface Builder Core Concepts	Nob Hill Wednesday 9:00AM

Labs

Xcode Lab	Tools Lab A Tuesday 2:00PM
Xcode Lab	Tools Lab C Wednesday 9:00AM
Xcode and Continuous Integration Lab	Tools Lab A Thursday 9:00AM
Xcode Lab	Tools Lab A Thursday 2:00PM
Xcode Lab	Tools Lab A Friday 9:00AM

WWDC2013