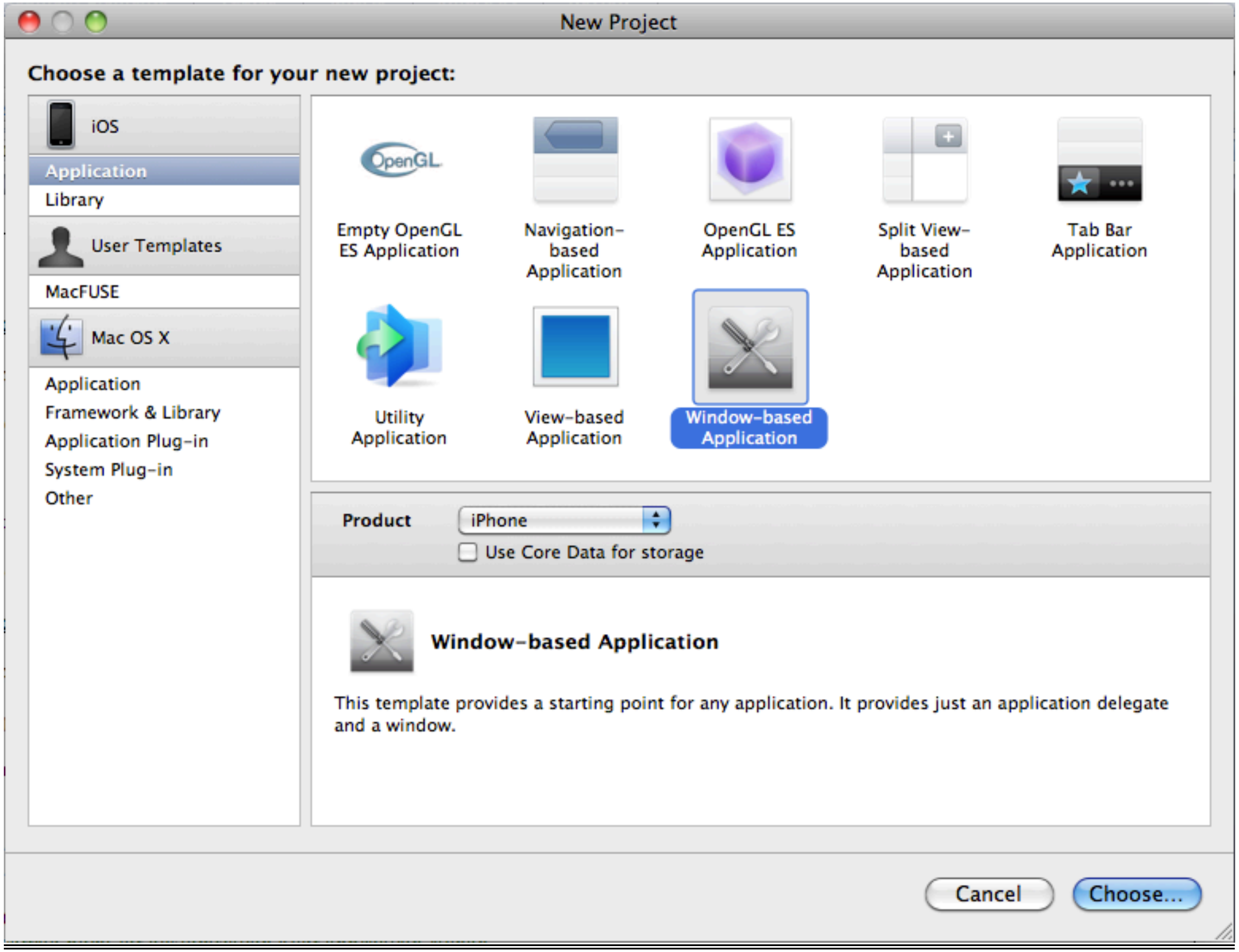
1. Window-based Application 선택



Choose a template for your new project:

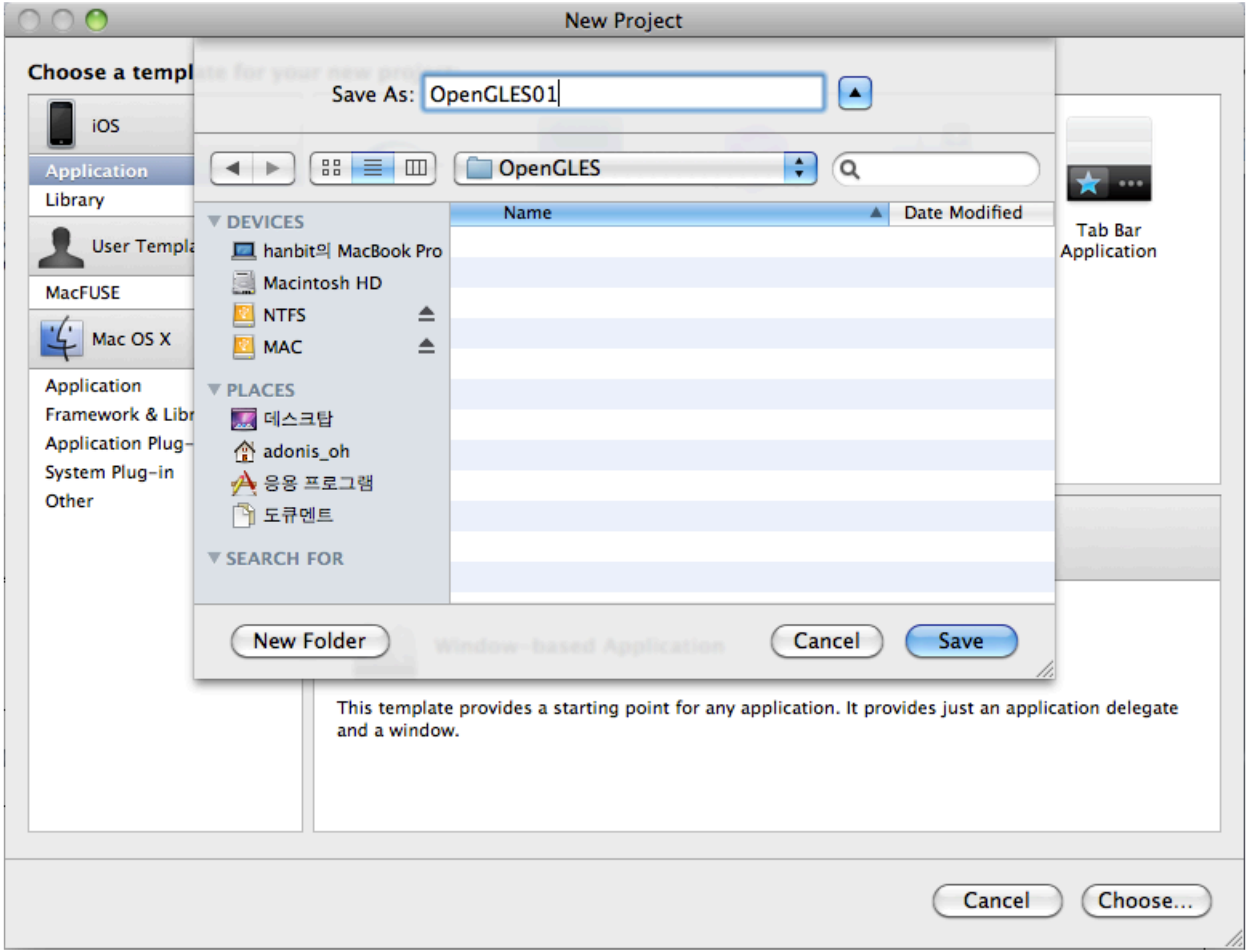
Product: iPhone

□ Use Core Data for storage

Window-based Application

This template provides a starting point for any application. It provides just an application delegate and a window.

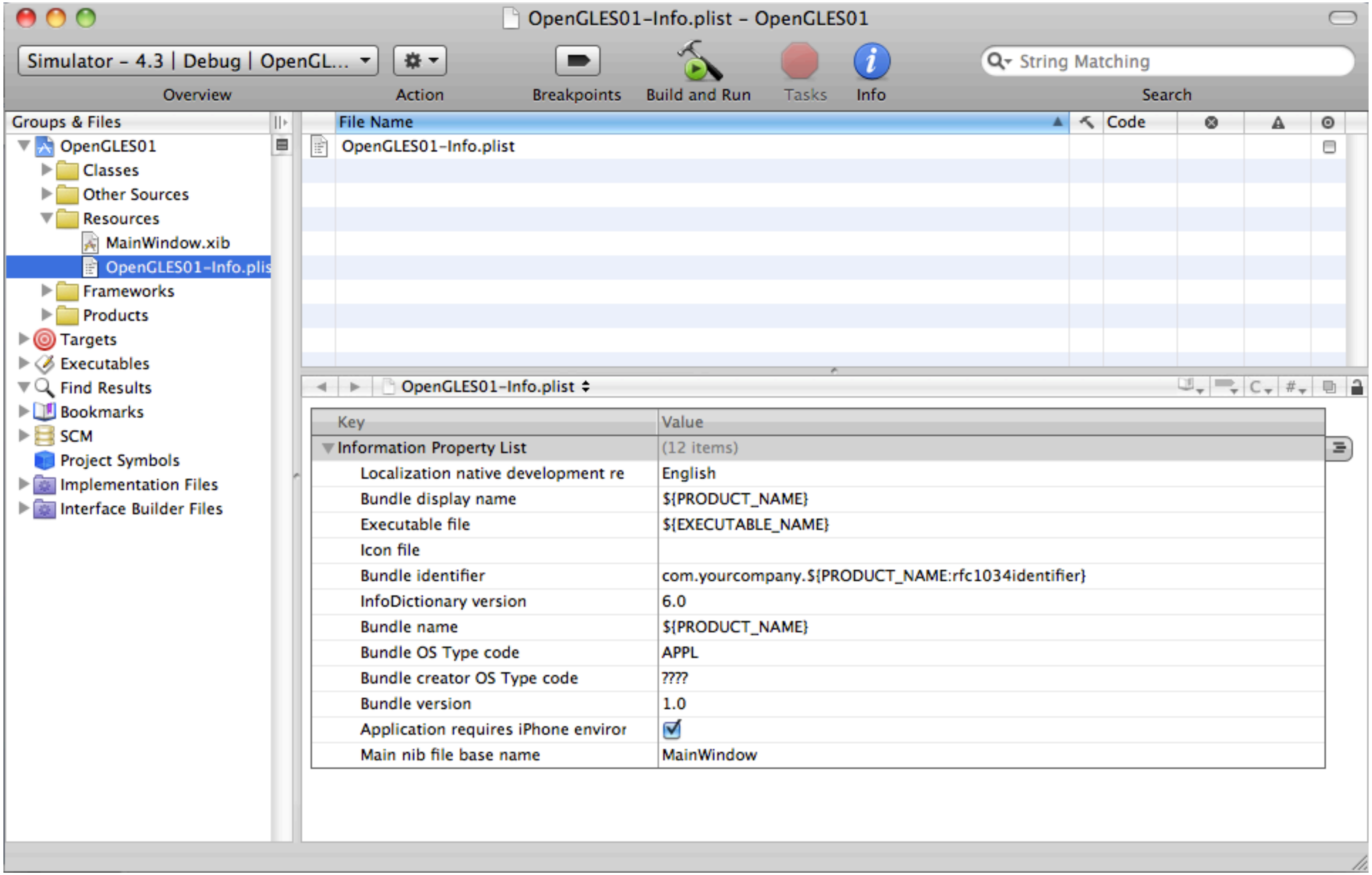
2. 프로젝트명 지정



Save As: OpenGLES01

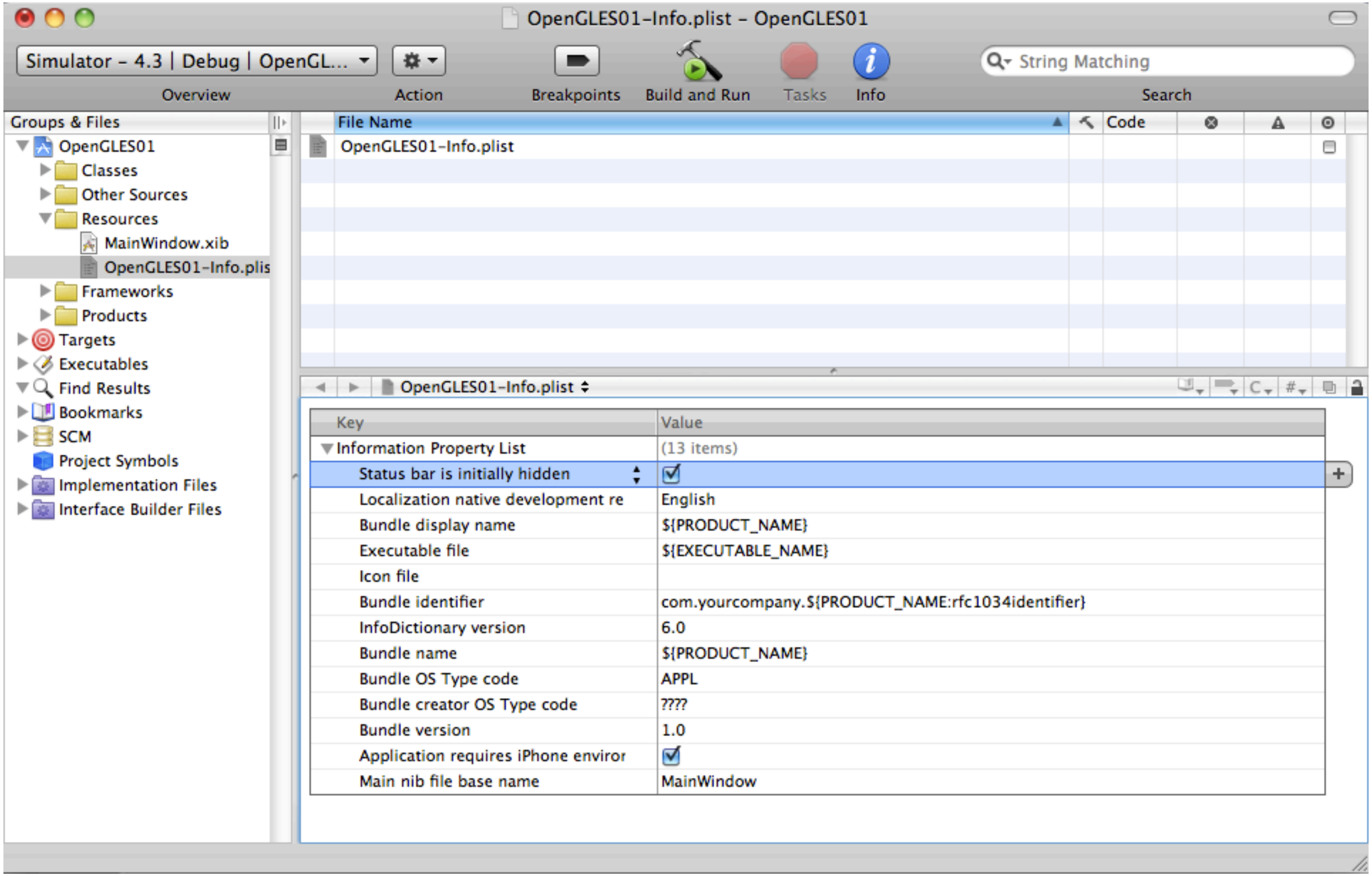
This template provides a starting point for any application. It provides just an application delegate and a window.

3. plist 파일 확인



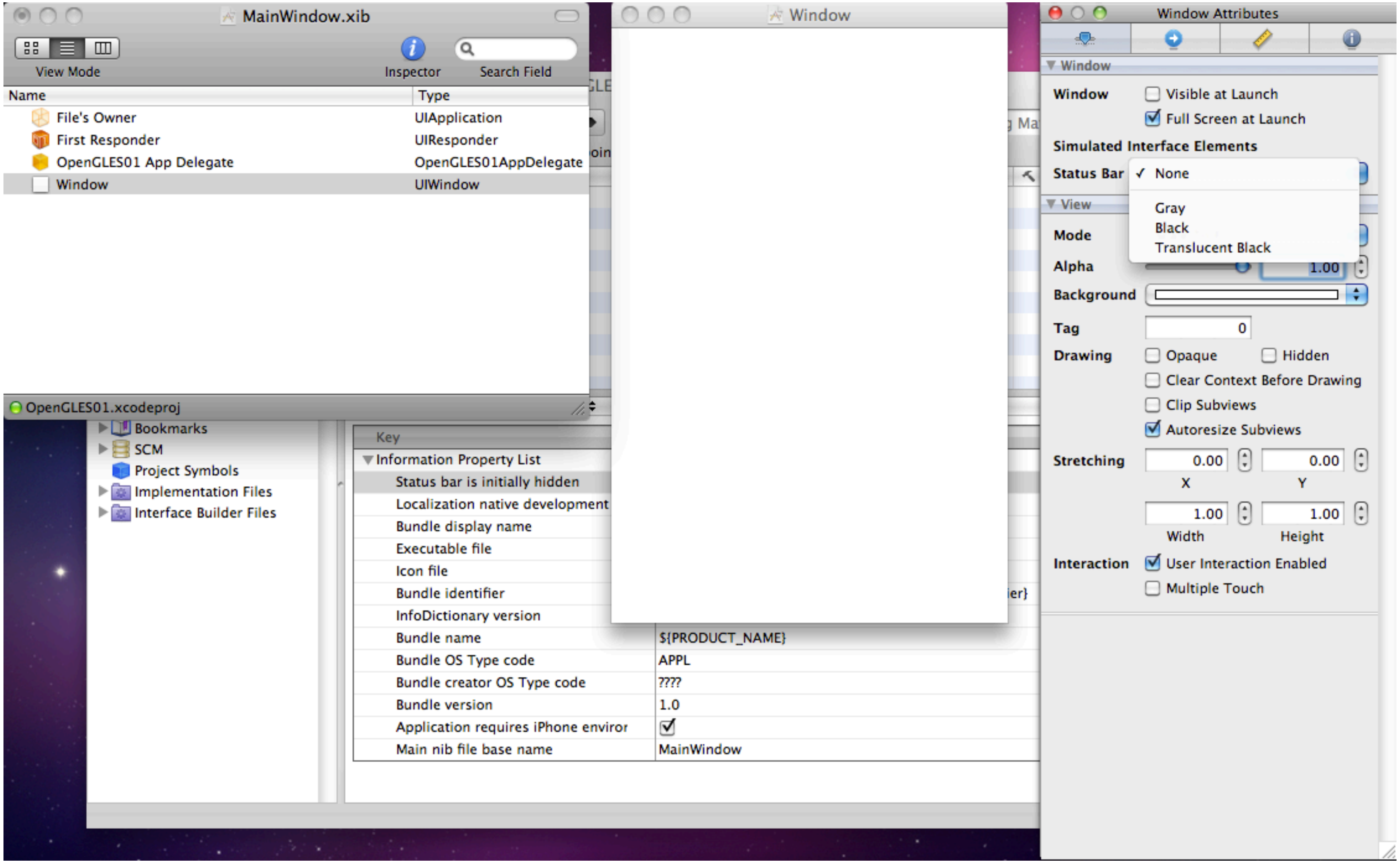
Key	Value
Information Property List	(12 items)
Localization native development re	English
Bundle display name	${PRODUCT_NAME}
Executable file	${EXECUTABLE_NAME}
Icon file	
Bundle identifier	com.yourcompany.${PRODUCT_NAME:rfc1034identifier}
InfoDictionary version	6.0
Bundle name	${PRODUCT_NAME}
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle version	1.0
Application requires iPhone enviror	☑
Main nib file base name	MainWindow

4. Status bar is initially hidden을 추가한 체크한 후 저장



Key	Value
Information Property List	(13 items)
Status bar is initially hidden	☑
Localization native development re	English
Bundle display name	${PRODUCT_NAME}
Executable file	${EXECUTABLE_NAME}
Icon file	
Bundle identifier	com.yourcompany.${PRODUCT_NAME:rfc1034identifier}
InfoDictionary version	6.0
Bundle name	${PRODUCT_NAME}
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle version	1.0
Application requires iPhone enviror	☑
Main nib file base name	MainWindow

5. MainWindow.xib 파일을 Interface Builder로 통해 연 후, Widow Attributes의 Status Bar 옵션을 None으로 선택해서 디자인상의 Status Bar를 제거



MainWindow.xib

View Mode | Inspector | Search Field

Name	Type
File's Owner	UIApplication
First Responder	UIResponder
OpenGLES01 App Delegate	OpenGLES01AppDelegate
Window	UIWindow

OpenGLES01.xcodeproj

- Bookmarks
- SCM
- Project Symbols
- Implementation Files
- Interface Builder Files

Key	
▼ Information Property List	
Status bar is initially hidden	
Localization native development	
Bundle display name	
Executable file	
Icon file	
Bundle identifier	ier}
InfoDictionary version	
Bundle name	${PRODUCT_NAME}
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle version	1.0
Application requires iPhone enviror	☑
Main nib file base name	MainWindow

Window

Window Attributes

▼ Window

Window: □ Visible at Launch; ☑ Full Screen at Launch

Simulated Interface Elements

Status Bar: ✓ None / Gray / Black / Translucent Black

▼ View

Mode

Alpha: 1.00

Background

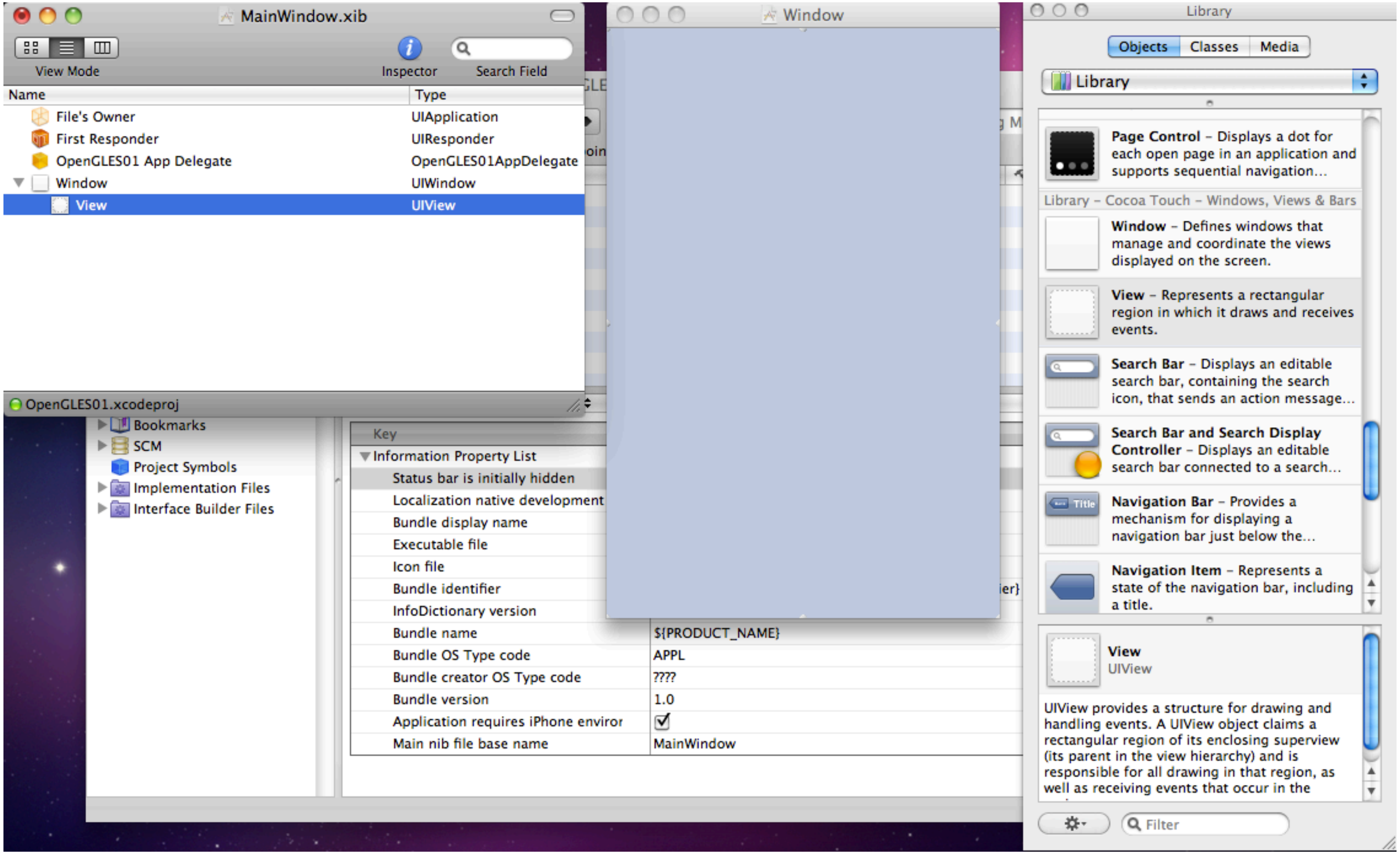
Tag: 0

Drawing: □ Opaque □ Hidden; □ Clear Context Before Drawing; □ Clip Subviews; ☑ Autoresize Subviews

Stretching: X 0.00, Y 0.00; Width 1.00, Height 1.00

Interaction: ☑ User Interaction Enabled; □ Multiple Touch

6. Library에서 View를 선택하여 Window 화면을 채운 후 저장하고 Interface Builder를 종료



Name	Type
File's Owner	UIApplication
First Responder	UIResponder
OpenGLES01 App Delegate	OpenGLES01AppDelegate
Window	UIWindow
View	UIView

Key	
Information Property List	
Status bar is initially hidden	
Localization native development	
Bundle display name	
Executable file	
Icon file	
Bundle identifier	ier}
InfoDictionary version	
Bundle name	${PRODUCT_NAME}
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle version	1.0
Application requires iPhone enviror	☑
Main nib file base name	MainWindow

Page Control – Displays a dot for each open page in an application and supports sequential navigation...

Window – Defines windows that manage and coordinate the views displayed on the screen.

View – Represents a rectangular region in which it draws and receives events.

Search Bar – Displays an editable search bar, containing the search icon, that sends an action message...

Search Bar and Search Display Controller – Displays an editable search bar connected to a search...

Navigation Bar – Provides a mechanism for displaying a navigation bar just below the...

Navigation Item – Represents a state of the navigation bar, including a title.

View
UIView

UIView provides a structure for drawing and handling events. A UIView object claims a rectangular region of its enclosing superview (its parent in the view hierarchy) and is responsible for all drawing in that region, as well as receiving events that occur in the

7. Delegate의 기본 구조

```
// OpenGLES10AppDelegate.h

#import <UIKit/UIKit.h>

@interface OpenGLES10AppDelegate : NSObject <UIApplicationDelegate> {
        UIWindow *window;
}

@property (nonatomic, retain) IBOutlet UIWindow *window;

@end

// OpenGLES10AppDelegate.m

#import "OpenGLES10AppDelegate.h"

@implementation OpenGLES10AppDelegate

@synthesize window;


#pragma mark -
#pragma mark Application lifecycle

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
// Override point for customization after application launch.
        [self.window makeKeyAndVisible];
        return YES;
```

```
}

- (void)applicationWillResignActive:(UIApplication *)application {
/*
Sent when the application is about to move from active to inactive state. This can occur for certain types of temporary interruptions
(such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the
background state.
Use this method to pause ongoing tasks, disable timers, and throttle down OpenGL ES frame rates. Games should use this method to
pause the game.
*/
}

- (void)applicationDidEnterBackground:(UIApplication *)application {
/*
Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to
restore your application to its current state in case it is terminated later.
If your application supports background execution, called instead of applicationWillTerminate: when the user quits.
*/
}

- (void)applicationWillEnterForeground:(UIApplication *)application {
/*
Called as part of   transition from the background to the inactive state: here you can undo many of the changes made on entering the
background.
*/
}

- (void)applicationDidBecomeActive:(UIApplication *)application {
/*
Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously in the
```

```
background, optionally refresh the user interface.
*/
}

- (void)applicationWillTerminate:(UIApplication *)application {
/*
Called when the application is about to terminate.
See also applicationDidEnterBackground:.
*/
}

#pragma mark -
#pragma mark Memory management

- (void)applicationDidReceiveMemoryWarning:(UIApplication *)application {
/*
Free up as much memory as possible by purging cached data objects that can be recreated (or reloaded from disk) later.
*/
}

- (void)dealloc {
	[window release];
	[super dealloc];
}

@end
```

8. OpenGL ES 1.1을 위한 설정

목차

1. framework 추가

Frameworks에 다음에 framework를 추가
QuartzCore.framework
OpenGLES.framwork
CoreGraphics.framework

2. EAGLView 생성

```
// EAGLView.h

#import <UIKit/UIKit.h>
#import <OpenGLES/EAGL.h>
#import <OpenGLES/ES1/gl.h>
#import <OpenGLES/ES1/glext.h>

/*
 This class wraps the CAEAGLLayer from CoreAnimation into a convenient UIView subclass.
 The view content is basically an EAGL surface you render your OpenGL scene into.
 Note that setting the view non-opaque will only work if the EAGL surface has an alpha channel.
 */

@interface EAGLView : UIView
```

```
{
@private

/* The pixel dimensions of the backbuffer */
        GLint backingWidth;
        GLint backingHeight;

        EAGLContext *context;

/* OpenGL names for the renderbuffer and framebuffers used to render to this view */
        GLuint viewRenderbuffer, viewFramebuffer;

/* OpenGL name for the depth buffer that is attached to viewFramebuffer, if it exists (0 if it does not exist) */
        GLuint depthRenderbuffer;

        BOOL animating;

/*
Use of the CADisplayLink class is the preferred method for controlling your animation timing.
CADisplayLink will link to the main display and fire every vsync when added to a given run-loop.
The NSTimer class is used only as fallback when running on a pre 3.1 device where CADisplayLink isn't available.
*/
        NSTimer *animationTimer;
        NSInteger animationFrameInterval;

        id displayLink;
        BOOL displayLinkSupported;
}

@property (readonly, nonatomic, getter=isAnimating) BOOL animating;
```

```
@property (nonatomic) NSInteger animationFrameInterval;

- (void)startAnimation;
- (void)stopAnimation;
- (void)drawView;

- (BOOL)createFramebuffer;
- (void)destroyFramebuffer;

- (void)checkGLError:(BOOL)visibleCheck;

@end

// EAGLView.m

#import <QuartzCore/QuartzCore.h>
#import <OpenGLES/EAGLDrawable.h>

#import "EAGLView.h"

@implementation EAGLView

@synthesize animating;
@synthesize animationFrameInterval;

// You must implement this method
+ (Class) layerClass
{
        return [CAEAGLLayer class];
}
```

```
// The GL view is stored in the nib file. When it's unarchived it's sent -initWithCoder:
- (id)initWithCoder:(NSCoder*)coder
{
	if((self = [super initWithCoder:coder])) {
// Get the layer
		CAEAGLLayer *eaglLayer = (CAEAGLLayer*) self.layer;
		
		eaglLayer.opaque = YES;
		eaglLayer.drawableProperties
		= [NSDictionary dictionaryWithObjectsAndKeys: [NSNumber numberWithBool:FALSE],
		    kEAGLDrawablePropertyRetainedBacking,
		    kEAGLColorFormatRGBA8,
		    kEAGLDrawablePropertyColorFormat,
		    nil];
		context = [[EAGLContext alloc] initWithAPI:kEAGLRenderingAPIOpenGLES1];
		
		if(!context || ![EAGLContext setCurrentContext:context] || ![self createFramebuffer]) {
			[self release];
			return nil;
		}
		
		animating = FALSE;
		displayLinkSupported = FALSE;
		animationFrameInterval = 1;
		displayLink = nil;
		animationTimer = nil;
		
// A system version of 3.1 or greater is required to use CADisplayLink. The NSTimer class is used as fallback when it isn't available.
		NSString *reqSysVer = @"3.1";
```

```
        NSString *currSysVer = [[UIDevice currentDevice] systemVersion];
        if ([currSysVer compare:reqSysVer options:NSNumericSearch] != NSOrderedAscending) {
            displayLinkSupported = TRUE;
        }
        [self drawView];
    }

    return self;
}

// Updates the OpenGL view when the timer fires
- (void)drawView
{
// Make sure that you are drawing to the current context
    [EAGLContext setCurrentContext:context];
    glBindFramebufferOES(GL_FRAMEBUFFER_OES, viewFramebuffer);

// This give us the size of the iPhone display
    CGRect rect = self.bounds;
    glViewport(0, 0, rect.size.width, rect.size.height);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    glClearColor(1.0f, 0.0f, 0.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glBindRenderbufferOES(GL_RENDERBUFFER_OES, viewRenderbuffer);
    [context presentRenderbuffer:GL_RENDERBUFFER_OES];
```

```
        [self checkGLError:NO];
}

- (void)layoutSubviews
{
        [EAGLContext setCurrentContext:context];
        [self destroyFramebuffer];
        [self createFramebuffer];
        [self drawView];
}

- (BOOL)createFramebuffer
{
        glGenFramebuffersOES(1, &viewFramebuffer);
        glGenRenderbuffersOES(1, &viewRenderbuffer);

        glBindFramebufferOES(GL_FRAMEBUFFER_OES, viewFramebuffer);
        glBindRenderbufferOES(GL_RENDERBUFFER_OES, viewRenderbuffer);
        [context renderbufferStorage:GL_RENDERBUFFER_OES fromDrawable:(id<EAGLDrawable>)self.layer];
        glFramebufferRenderbufferOES(
                                                        GL_FRAMEBUFFER_OES,
                                                        GL_COLOR_ATTACHMENT0_OES,
                                                        GL_RENDERBUFFER_OES,
                                                        viewRenderbuffer);

        glGetRenderbufferParameterivOES(GL_RENDERBUFFER_OES, GL_RENDERBUFFER_WIDTH_OES, &backingWidth);
        glGetRenderbufferParameterivOES(GL_RENDERBUFFER_OES, GL_RENDERBUFFER_HEIGHT_OES, &backingHeight);

        if(glCheckFramebufferStatusOES(GL_FRAMEBUFFER_OES) != GL_FRAMEBUFFER_COMPLETE_OES) {
                NSLog(@"failed to make complete framebuffer object %x", glCheckFramebufferStatusOES(GL_FRAMEBUFFER_OES));
```

```
		return NO;
	}

	return YES;
}

- (void)destroyFramebuffer
{
	glDeleteFramebuffersOES(1, &viewFramebuffer);
	viewFramebuffer = 0;
	glDeleteRenderbuffersOES(1, &viewRenderbuffer);
	viewRenderbuffer = 0;

	if(depthRenderbuffer) {
		glDeleteRenderbuffersOES(1, &depthRenderbuffer);
		depthRenderbuffer = 0;
	}
}

- (NSInteger) animationFrameInterval
{
	return animationFrameInterval;
}

- (void) setAnimationFrameInterval:(NSInteger)frameInterval
{
/*
Frame interval defines how many display frames must pass between each time the display link fires. The display link will only fire 30
times a second when the frame internal is two on a display that refreshes 60 times a second. The default frame interval setting of one
will fire 60 times a second when the display refreshes at 60 times a second. A frame interval setting of less than one results in
```

```
undefined behavior.
*/
        if (frameInterval >= 1)
        {
                animationFrameInterval = frameInterval;

                if (animating)
                {
                        [self stopAnimation];
                        [self startAnimation];
                }
        }
}

- (void) startAnimation
{
        if (!animating)
        {
                if (displayLinkSupported)
                {
/*
CADisplayLink is API new to iPhone SDK 3.1. Compiling against earlier versions will result in a warning, but can be dismissed if the
system version runtime check for CADisplayLink exists in -initWithCoder:. The runtime check ensures this code will not be called in
system versions earlier than 3.1.
*/
                        displayLink = [NSClassFromString(@"CADisplayLink") displayLinkWithTarget:self selector:@selector(drawView)];
                        [displayLink setFrameInterval:animationFrameInterval];
                        [displayLink addToRunLoop:[NSRunLoop currentRunLoop] forMode:NSDefaultRunLoopMode];
                } else {
                        animationTimer = [NSTimer scheduledTimerWithTimeInterval:(NSTimeInterval)((1.0 / 60.0) * animationFrameInterval)
```

```
                            target:self
                            selector:@selector(drawView)
                            userInfo:nil
                            repeats:TRUE];
        }
        animating = TRUE;
    }
}

- (void)stopAnimation
{
    if (animating)
    {
        if (displayLinkSupported)
        {
            [displayLink invalidate];
            displayLink = nil;
        } else {
            [animationTimer invalidate];
            animationTimer = nil;
        }

        animating = FALSE;
    }
}

- (void)checkGLError:(BOOL)visibleCheck {
   GLenum error = glGetError();

   switch (error) {
        case GL_INVALID_ENUM:
            NSLog(@"GL Error: Enum argument is out of range");
            break;
        case GL_INVALID_VALUE:
            NSLog(@"GL Error: Numeric value is out of range");
            break;
        case GL_INVALID_OPERATION:
            NSLog(@"GL Error: Operation illegal in current state");
            break;
        case GL_STACK_OVERFLOW:
            NSLog(@"GL Error: Command would cause a stack overflow");
            break;
        case GL_STACK_UNDERFLOW:
            NSLog(@"GL Error: Command would cause a stack underflow");
            break;
        case GL_OUT_OF_MEMORY:
            NSLog(@"GL Error: Not enough memory to execute command");
            break;
        case GL_NO_ERROR:
            if (visibleCheck) {
                NSLog(@"No GL Error");
            }
            break;
        default:
            NSLog(@"Unknown GL Error");
            break;
    }
}

// Release resources when they are no longer needed.
```

```
- (void)dealloc
{
    if([EAGLContext currentContext] == context) {
        [EAGLContext setCurrentContext:nil];
    }

    [context release];
    context = nil;

    [super dealloc];
}

@end
```

```
// OpenGLESAppDelegate.m

#import "OpenGLESAppDelegate.h"
#import "EAGLView.h"

@implementation OpenGLESAppDelegate

@synthesize window;
@synthesize glView;

#pragma mark -
#pragma mark Application lifecycle

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [glView startAnimation];
    [self.window makeKeyAndVisible];
    return YES;
}

- (void)applicationWillResignActive:(UIApplication *)application {
    [glView stopAnimation];
}

- (void)applicationDidEnterBackground:(UIApplication *)application {

}

- (void)applicationWillEnterForeground:(UIApplication *)application {
}

- (void)applicationDidBecomeActive:(UIApplication *)application {
    [glView startAnimation];
}

- (void)applicationWillTerminate:(UIApplication *)application {
    [glView stopAnimation];
}


#pragma mark -
#pragma mark Memory management

- (void)applicationDidReceiveMemoryWarning:(UIApplication *)application {
}
```

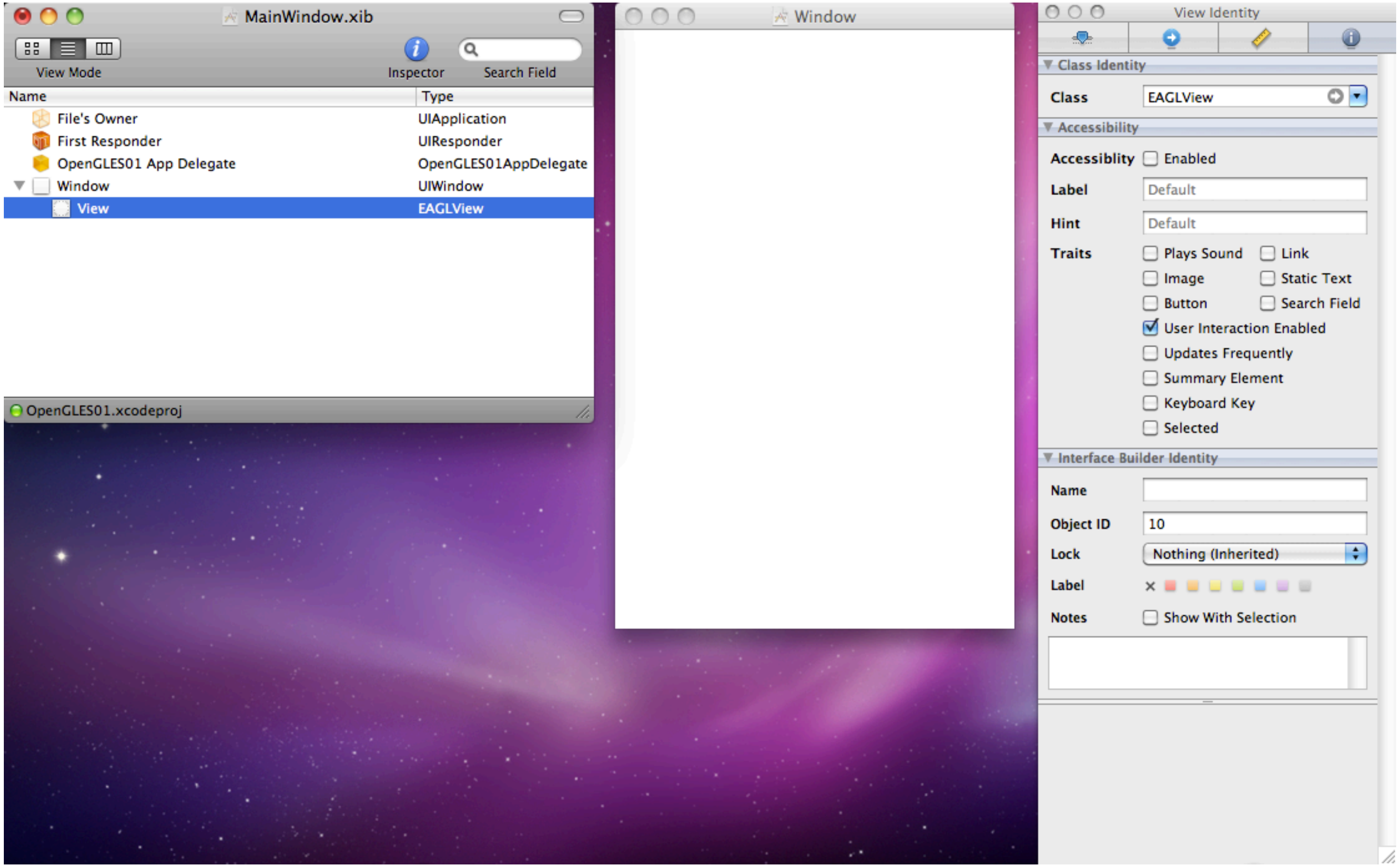
```
- (void)dealloc {
	[window release];
	[glView release];

	[super dealloc];
}

@end
```

4. xib 수정

1. UIViwe Type을 EAGLView 클래스로 수정



Name	Type
File's Owner	UIApplication
First Responder	UIResponder
OpenGLES01 App Delegate	OpenGLES01AppDelegate
Window	UIWindow
View	EAGLView

Class Identity

Class: EAGLView

Accessibility

Accessiblity ☐ Enabled

Label: Default

Hint: Default

Traits
☐ Plays Sound ☐ Link
☐ Image ☐ Static Text
☐ Button ☐ Search Field
☑ User Interaction Enabled
☐ Updates Frequently
☐ Summary Element
☐ Keyboard Key
☐ Selected

Interface Builder Identity

Name:

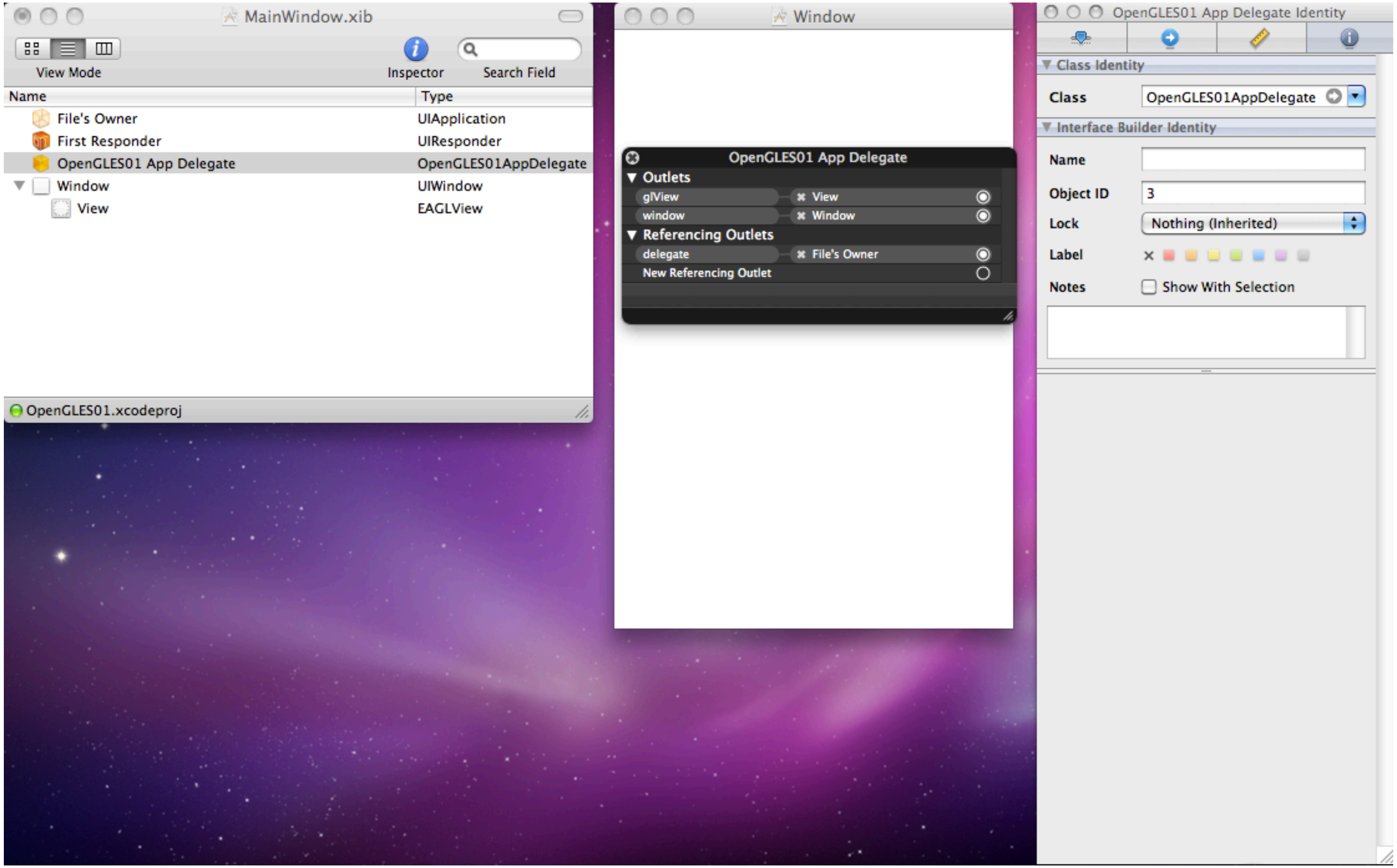
Object ID: 10

Lock: Nothing (Inherited)

Label: ×

Notes ☐ Show With Selection

2. OpenGLES01 App Delegate에 glView를 EAGLView와 연결



Name	Type
File's Owner	UIApplication
First Responder	UIResponder
OpenGLES01 App Delegate	OpenGLES01AppDelegate
Window	UIWindow
View	EAGLView

OpenGLES01.xcodeproj

OpenGLES01 App Delegate

Outlets
- glView — View
- window — Window

Referencing Outlets
- delegate — File's Owner
- New Referencing Outlet

OpenGLES01 App Delegate Identity

Class Identity

Class: OpenGLES01AppDelegate

Interface Builder Identity

Name:

Object ID: 3

Lock: Nothing (Inherited)

Label:

Notes: □ Show With Selection

// 변경 1

- drawView의 반복적 호출

```
- (void)drawView
{
        [EAGLContext setCurrentContext:context];
        glBindFramebufferOES(GL_FRAMEBUFFER_OES, viewFramebuffer);

        CGRect rect = self.bounds;
        glViewport(0, 0, rect.size.width, rect.size.height);

        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();

        glClearColor(1.0f, 0.0f, 0.0f, 1.0f);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        NSLog(@"drawView 호출");

        glBindRenderbufferOES(GL_RENDERBUFFER_OES, viewRenderbuffer);
        [context presentRenderbuffer:GL_RENDERBUFFER_OES];

        [self checkGLError:NO];
}
```