Quick Look for Custom Types in the Xcode Debugger

Developer

Contents

Figures

About Variables Quick Look for Custom Types

Important: This is a preliminary document for an API or technology in development. Apple is supplying this information to help you plan for the adoption of the technologies and programming interfaces described herein for use on Apple-branded products. This information is subject to change, and software implemented according to this document should be tested with final operating system software and final documentation. Newer versions of this document may be provided with future betas of the API or technology.

The debugger in Xcode includes the variables Quick Look feature, a way to view variables in your app by displaying their contents graphically in a pop-up display. Quick Look lets you display a graphical rendering of object contents by pressing the Space bar with a variable selected in the debugger variables view.

At a Glance

The Xcode 5 debugger gave you the ability to dynamically visualize selected operating system class types with the variables Quick Look feature. This document addresses two important enhancements to this capability that were introduced with Xcode 5.1:

- Variables instantiated from your custom types can render a Quick Look display.
- Additional operating system classes now support Quick Look display.

Together these two enhancements provide flexibilty in how you display your variables when debugging. To provide your custom class with a Quick Look display in the debugger, you provide a method for Quick Look to use in the object class. The method, called by Xcode, returns an object with a type matching one of the operating system classes supported by Quick Look display.

A customized Quick Look method for each of your custom object classes can be implemented, each rendering a live representation of the underlying variable in the way that makes best sense for the type. For example, a `Person` object could show an image of the person, or perhaps a map with a pin on a person's home address, whichever is relevant to the implementation of the custom class in your context. You choose the operating system object type to return depending upon what best fits a rendering of your custom class objects.

See Also

Apple provides the following video presentations that show more about using the Xcode debugger:

- WWDC 2014: Debugging with Xcode 6: Learn how apps enqueue work, explore and fix user interfaces, add custom Quick Look support.
- WWDC 2013: Debugging with Xcode: Detect and fix performance problems using the Xcode graphical debugger.
- WWDC 2013: Advanced Debugging with LLDB: Debug using Terminal and the Xcode graphical debugger.

Enabling Quick Look for Custom Types

The variables Quick Look feature in the Xcode debugger allows you to obtain a quick visual assessment of the state of an object variable through a graphical rendering, displayed in a popover window either in the debugger variables view or in place in your source code. For supported operating system types, Quick Look displays debugger Quick Look object primitives to service this visualization.

This chapter describes how you implement a Quick Look method for your custom class types so that object variables of those types can also be rendered visually in the Quick Look popover window.

Implement the Quick Look method

For your custom class type, implement a method named `debugQuickLookObject`.

```
- (id)debugQuickLookObject
{
    // allocate the return object for the data you wish to represent
    //   Note: "NSImage" is used here arbitrarily for illustration purposes.
    NSImage *_quickLookImage = [...]

    // code that draws a representation of the variable state
    // ...

    // return the object
    return _quickLookImage;
}
```

A set of the most common operating system types useful for rendering visualizations are able to output debugger Quick Look object primitives which Xcode uses for the Quick Look display. Your `debugQuickLookObject` method must return an object of one of these types. See Operating System Types Supporting debugQuickLookObject (page 10) for a table of operating system types and examples of their standard Quick Look displays so that you can choose the most appropriate one to render your custom type.

When you select a variable of the custom class type in the Xcode debugger variables view and use Quick Look, the `debugQuickLookObject` method defined for the class is called. The Quick Look popover display presents the returned object.

Caveats

Because your `debugQuickLookObject` method runs in the debugger, when you are inside your paused application, it is best to be economical in your method implementation. Write as little code as needed to represent your variable's state as a useful visual graphic. Remember that running code when in a paused application can have side effects. If possible, cache the item you want to return.

Sample implementation

The following `debugQuickLookObject` method demonstrates a trivially defined custom type named `MyCustomClass` used with Quick Look in the debugger. The object type is declared as:

```
#import <Foundation/Foundation.h>

@interface MyCustomClass : NSObject

@end
```

The `debugQuickLookObject` method in `MyCustomClass.m` draws a graphic for Quick Look to display and returns it in an `NSImage` object.

```
- (id)debugQuickLookObject

{

    // Note that for simplicity in presentation, this sample allocates and returns
 an NSImage object.

    // Using a cached object instead is preferred; it minimizes potential impact
on the paused app context.

    NSImage *image = [NSImage imageWithSize:QUICK_LOOK_IMAGE_SIZE flipped:NO
drawingHandler:^BOOL(NSRect dstRect) {


        CGFloat midX = NSMidX(dstRect);

        CGFloat midY = NSMidY(dstRect);

        NSUInteger numCircles = 5;

        CGFloat circleSpacing = -10.0;
```

```
        CGFloat circleRadius = 20.0;
        CGFloat circleDiameter = circleRadius * 2;
        CGFloat stride = circleDiameter + circleSpacing;
        CGFloat currentCircleX = midX - (stride * numCircles)/2.0;


        NSColor *strokeColor = [NSColor colorWithCalibratedRed:0.10 green:0.41
blue:1.0 alpha:1.0];
        NSColor *fillColor = [strokeColor colorWithAlphaComponent:0.15];


        for (NSUInteger i=0; i < numCircles; i++) {
            NSRect circleRect = NSMakeRect(currentCircleX, midY - circleRadius,
circleDiameter, circleDiameter);
            NSBezierPath *circlePath = [NSBezierPath
bezierPathWithOvalInRect:circleRect];
            [fillColor set];
            [circlePath fill];
            [strokeColor set];
            [circlePath stroke];
            currentCircleX += stride;
        }
        return YES;
    }];
    return image;
}
```

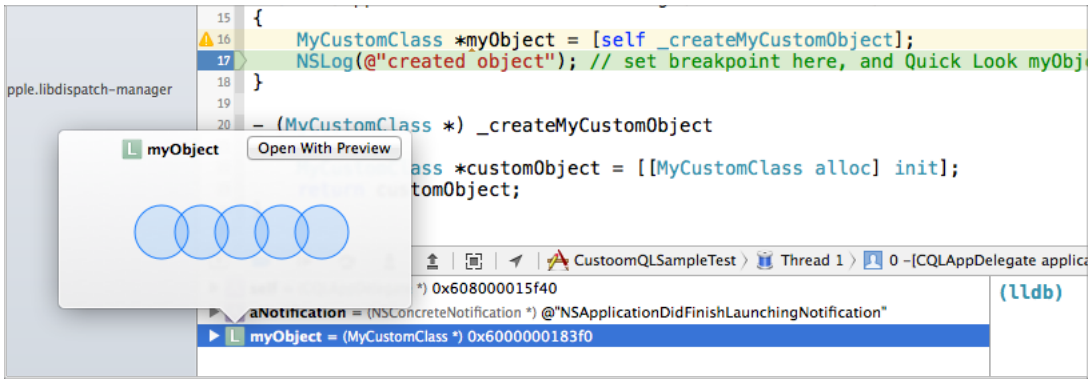
To see the `debugQuickLook` method work in the Xcode debugger, add the`MyCustomClass.h` interface and `MyCustomClass.m` implementation files to a simple app created from a template. Inser the following code into the `AppDelegate.m` file. A breakpoint set at the `NSLog()` call allows you to Quick Look the returned `myObject` custom class variable.

```
- (void)applicationDidFinishLaunching:(NSNotification *)aNotification
{
    MyCustomClass *myObject = [self _createMyCustomObject];
    NSLog(@"created object"); // set breakpoint here, and Quick Look myObject
}
```

```
- (MyCustomClass *) _createMyCustomObject
{
    MyCustomClass *customObject = [[MyCustomClass alloc] init];
    return customObject;
}
```

The resulting Quick Look display is shown in Figure 1-1.

Figure 1-1 Quick Look displaying custom type



Operating System Types Supporting debugQuickLookObject

Your custom Quick Look display method, `debugQuickLookObject` (see Implement the Quick Look method (page 6)), must return an object whose type is one of the operating system classes that supports rendering custom classes in Quick Look. The following table lists these iOS and OS X system classes, grouped by similar content.

> **Important:** Many operating system types are viewable with the Xcode debugger Quick Look tool. Only the ones listed here can be used as a return type from your `debugQuickLookObject` method.

The examples in Quick Look Examples for System Classes (page 12) illustrate the standard Quick Look displays for these operating system object types.

Returnable Types for debugQuickLookObject

Object Type	Example	Implementation Notes
Default	See the figure in Default (page 12).	No custom Quick Look display available.

Image classes:
`NSImage`
`UIImage`
`NSImageView`
`UIImageView`
`CIImage`
`NSBitmapImageRep`
	See the figure in Image Class Types (page 13).	For `CIImage`, the Xcode target must be OS X Mavericks or later.
		
		`NSBitmapImageRep` is supported in Xcode 5.1 or later.

Cursor class: `NSCursor`	See the figure in Cursor Class Type (page 13).	

Object Type	Example	Implementation Notes

Color classes:
`NSColor`
`UIColor`
	See the figure in Color Class Types (page 14).	

BezierPath classes:
`NSBezierPath`
`UIBezierPath`
	See the figure in BezierPath Class Types (page 14).	

Location classes: `CLLocation`	See the figure in Location Class Type (page 15).	

View classes:
`NSView`
`UIView`
	See the figure in View Class Types (page 16).	`UIView` supported in Xcode 5.1 or later.

String class: `NSString`	See the figure in String Class Type (page 17).	
AttributedString class: `NSAttributedString`	See the figure in AttributedString Class Type (page 18).	
Data class: `NSData`	See the figure in Data Class Type (page 19).	
URL class: `NSURL`	See the figure in URL Class Type (page 20).	

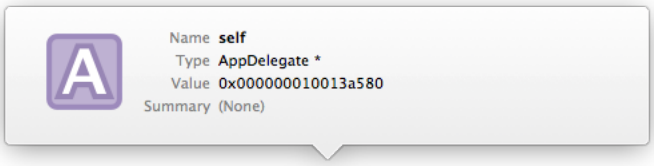
Object Type	Example	Implementation Notes

SpriteKit classes:
`SKSpriteNode`
`SKShapeNode`
`SKTexture`
`SKTextureAtlas`
	See the figure in SpriteKit Class Types (page 21).	SpriteKit classes are supported in Xcode 6 or later. The target must be iOS 8 or OS X Yosemite, or later.

Quick Look Examples for System Classes

Default

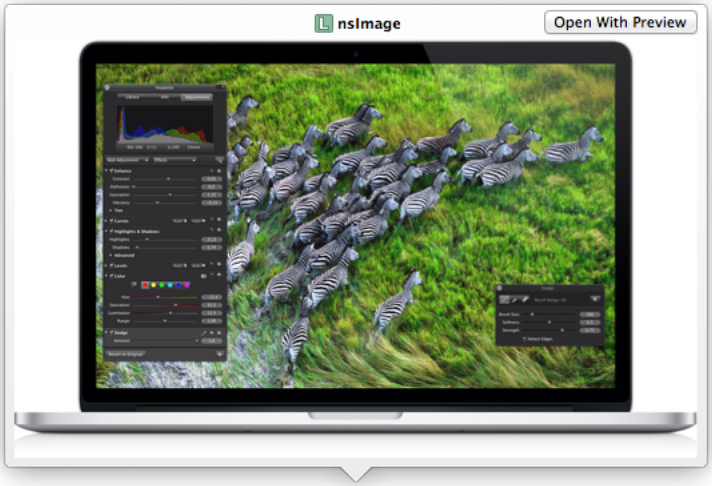
The default Quick Look display is used when no Quick Look debug object is available.



(*Return to table:* Returnable Types for debugQuickLookObject (page 10))

Image Class Types

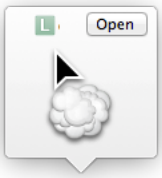
Image class types open in a Quick Look display with a button that allows you to open the rendering graphic in Preview.

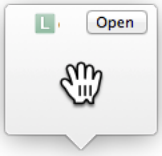


(*Return to table:* Returnable Types for debugQuickLookObject (page 10))

Cursor Class Type

Cursor class types open into a Quick Look display with a button that allows you to view them in Preview.

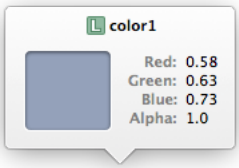




(*Return to table:* Returnable Types for debugQuickLookObject (page 10))

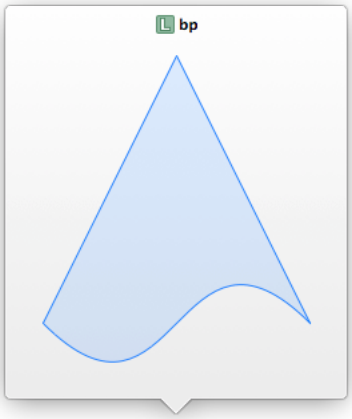
Color Class Types

Color class types show a sample swatch with red, green, blue, and alpha channel values represented in the range from 0.0 to 1.0.



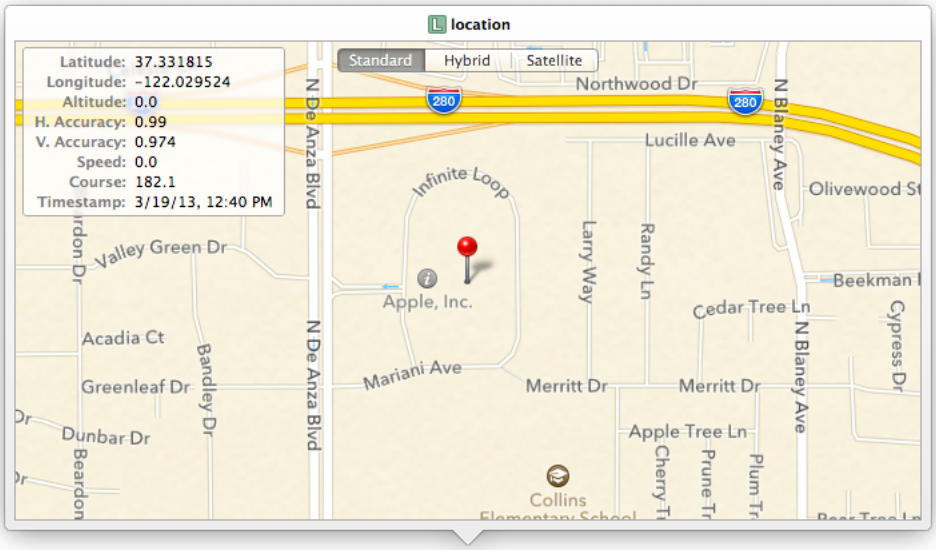
(*Return to table:* Returnable Types for debugQuickLookObject (page 10))

BezierPath Class Types



(*Return to table:* Returnable Types for debugQuickLookObject (page 10))

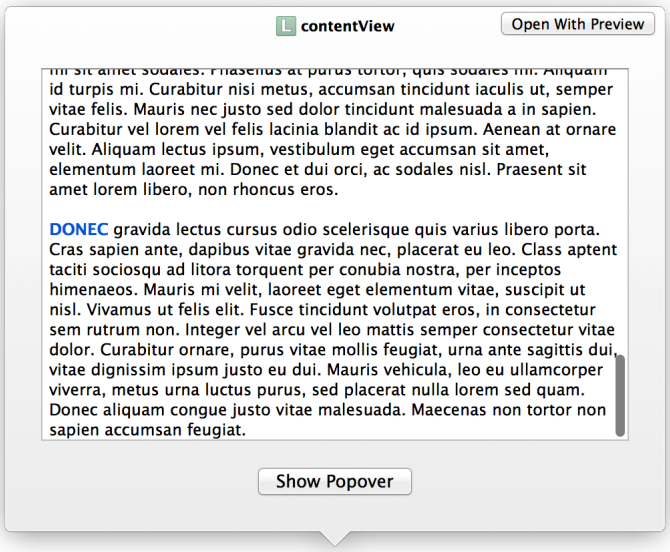
Location Class Type

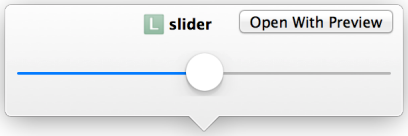


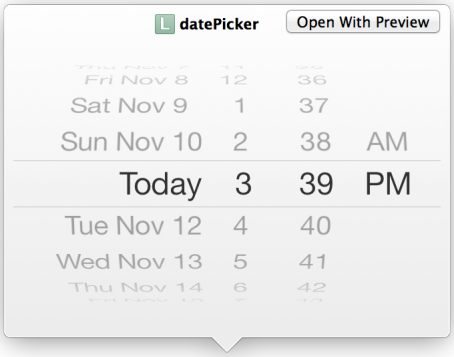
(*Return to table:* Returnable Types for debugQuickLookObject (page 10))

View Class Types

View class types open in Quick Look displays with buttons that allow you to open the graphic displayed in Preview.



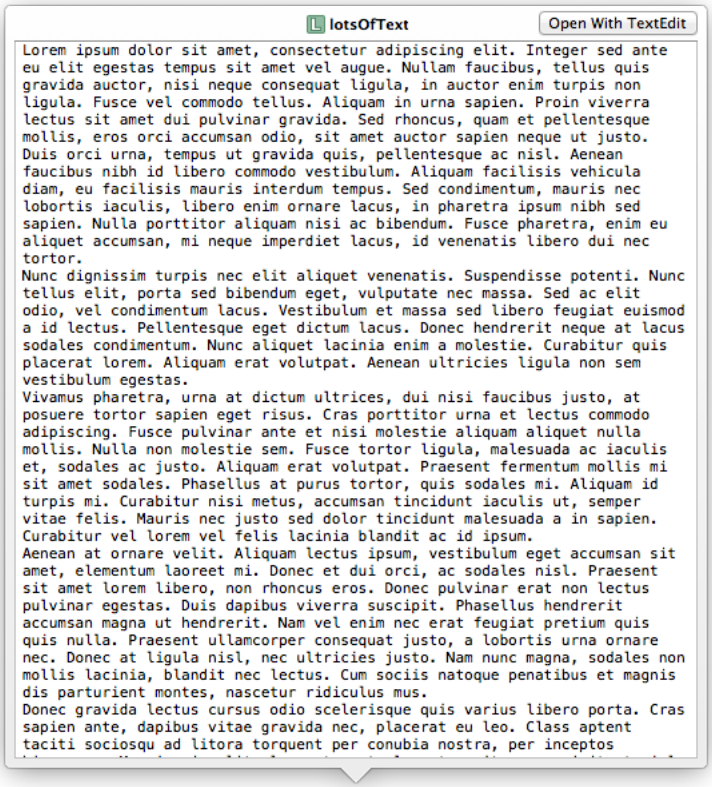




(*Return to table:* Returnable Types for debugQuickLookObject (page 10))

String Class Type

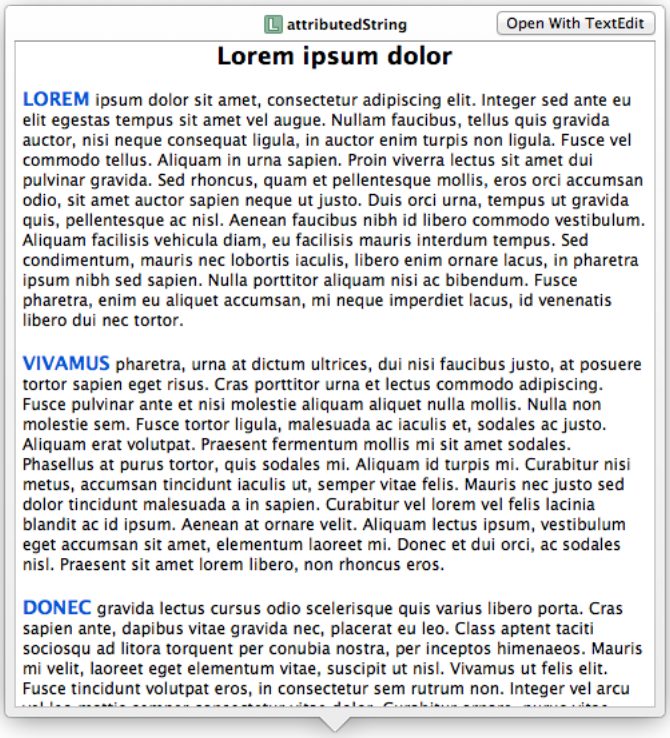
String class types open in a view equipped with a button that transfers the contents into TextEdit.



(*Return to table:* Returnable Types for debugQuickLookObject (page 10))

AttributedString Class Type

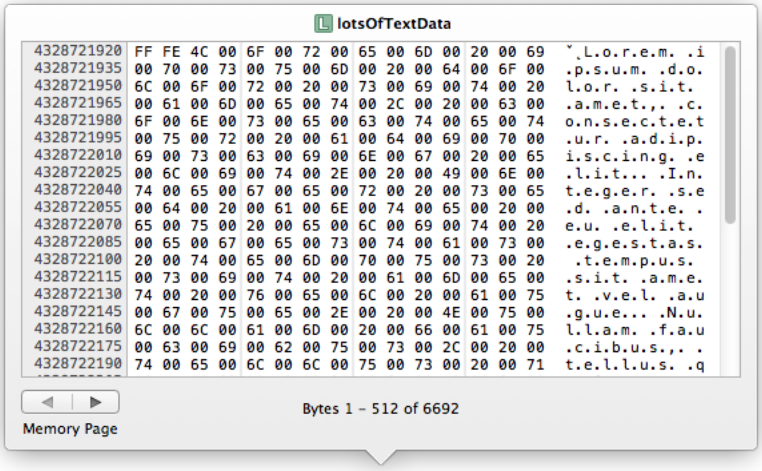
AttributedString class types open in a view equipped with a button that transfers the contents into TextEdit.



(*Return to table:* Returnable Types for debugQuickLookObject (page 10))

Data Class Type

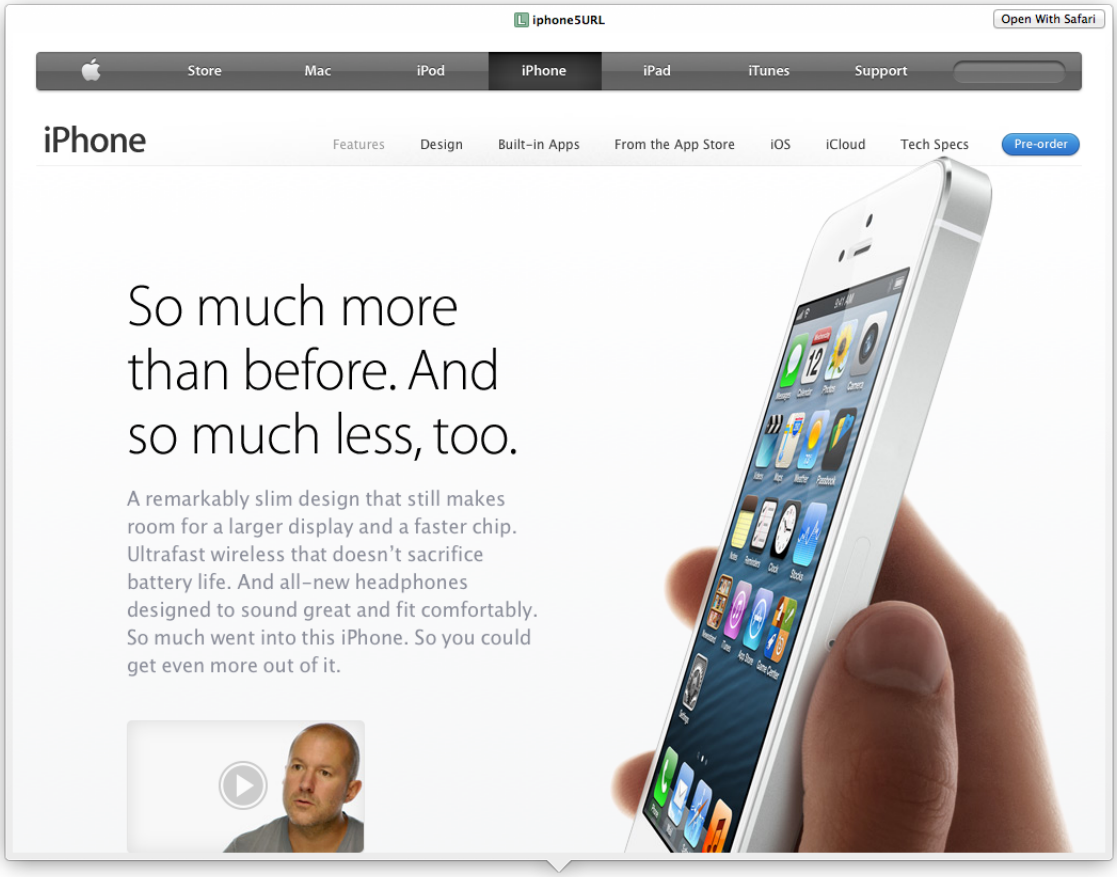
The `NSData` class type opens in a three-column view of offset, hexadecimal, and ASCII values.



(*Return to table:* Returnable Types for debugQuickLookObject (page 10))

URL Class Type

This example of an NSURL Quick Look is displayed from a web source.



(*Return to table:* Returnable Types for debugQuickLookObject (page 10))

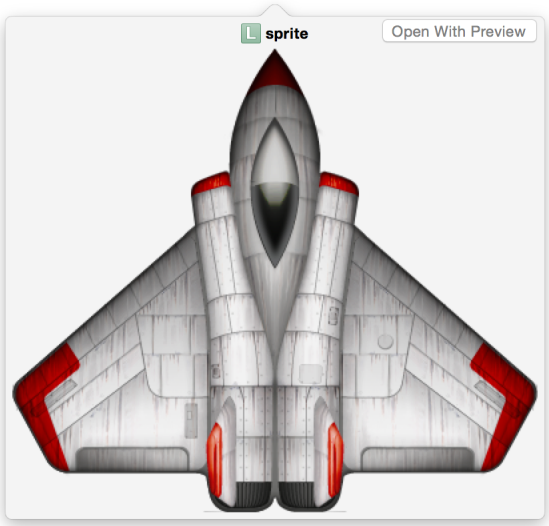
This example of an NSURL Quick Look is displayed from a local or system Quick Look source.

(*Return to table:* Returnable Types for debugQuickLookObject (page 10))

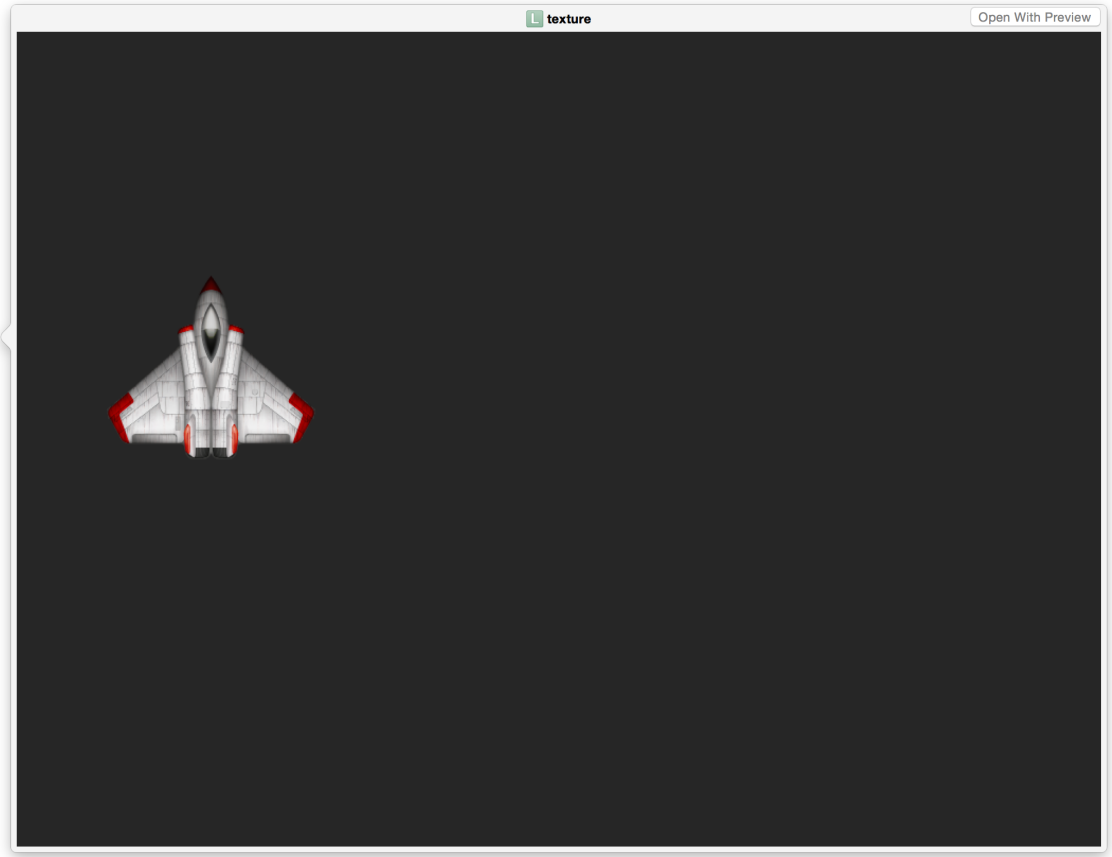
SpriteKit Class Types

SpriteKit class types open in a Quick Look display with a button that allows you to open the rendered graphic in Preview.

Example of an `SKSpriteNode`:



Example of an `SKTexture`:



(*Return to table:* Returnable Types for debugQuickLookObject (page 10))

Document Revision History

This table describes the changes to *Quick Look for Custom Types in the Xcode Debugger*.

Date	Notes
2014-09-17	Updated to include new Quick Look object return types.
2014-03-10	New document that shows how to enable Quick Look for custom object types with the Xcode debugger.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010