

---

# NSOpenGLContext Class Reference

Graphics & Animation: 3D Drawing



2009-08-13



Apple Inc.  
© 2009 Apple Inc.  
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, Cocoa, Mac, Mac OS, and Quartz are trademarks of Apple Inc., registered in the United States and other countries.

Finder is a trademark of Apple Inc.

OpenGL is a registered trademark of Silicon Graphics, Inc.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE**

**ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# Contents

## NSOpenGLContext Class Reference 5

---

Overview	5
Tasks	5
Context Creation	5
Managing the Current Context	6
Drawable Object Management	6
Flushing the Drawing Buffer	6
Copying Attributes	6
Context Parameter Handling	6
Working with Virtual Screens	7
Creating Textures	7
Getting the CGL Context Object	7
Working with Pixel Buffers	7
Class Methods	7
clearCurrentContext	7
currentContext	8
Instance Methods	8
CGLContextObj	8
clearDrawable	9
copyAttributesFromContext:withMask:	9
createTexture:fromView:internalFormat:	10
currentVirtualScreen	10
flushBuffer	11
getValues:forParameter:	12
initWithCGLContextObj:	12
initWithFormat:shareContext:	12
makeCurrentContext	13
pixelBuffer	14
pixelBufferCubeMapFace	14
pixelBufferMipMapLevel	15
setCurrentVirtualScreen:	15
setFullScreen	16
setOffScreen:width:height:rowbytes:	16
setPixelBuffer:cubeMapFace:mipMapLevel:currentVirtualScreen:	17
setTextureImageToPixelBuffer:colorBuffer:	18
setValues:forParameter:	19
setView:	19
update	20
view	20
Constants	21
NSOpenGLContextParameter	21

[Document Revision History](#) 23

---

[Index](#) 25

---

# NSOpenGLContext Class Reference

---

<b>Inherits from</b>	NSObject
<b>Conforms to</b>	NSObject (NSObject)
<b>Framework</b>	/System/Library/Frameworks/AppKit.framework
<b>Availability</b>	Available in Mac OS X v10.0 and later.
<b>Companion guide</b>	Cocoa Drawing Guide
<b>Declared in</b>	NSOpenGL.h
<b>Related sample code</b>	GLUT LiveVideoMixer2 OpenGLCaptureToMovie Quartz Composer Live DV Quartz Composer Texture

## Overview

All OpenGL calls are rendered into an OpenGL graphics context, which in Cocoa is represented by the `NSOpenGLContext` class. The context is created using an `NSOpenGLPixelFormatObject` that specifies the context's buffer types and other attributes. A context can be full-screen, offscreen, or associated with an `NSView` object. A context draws into its **drawable object**, which is the frame buffer that is the target of OpenGL drawing operations.

Every `NSOpenGLContext` object wraps a low-level, platform-specific Core OpenGL (CGL) context. Your application can retrieve the CGL context by calling the `CGLContextObj` (page 8) method. For more information on the underlying CGL context, see *CGL Reference*.

## Tasks

### Context Creation

- `initWithFormat:shareContext:` (page 12)  
Returns an `NSOpenGLContext` object initialized with the specified pixel format information.
- `initWithCGLContextObj:` (page 12)  
Initializes and returns a `NSOpenGLContext` object using an existing CGL context.

## Managing the Current Context

- + [clearCurrentContext](#) (page 7)  
Sets the current context to `nil`.
- + [currentContext](#) (page 8)  
Returns the current OpenGL graphics context.
- [makeCurrentContext](#) (page 13)  
Sets the receiver as the current OpenGL context object.

## Drawable Object Management

- [setView:](#) (page 19)  
Sets the receiver's viewport to the specified `NSView` object.
- [view](#) (page 20)  
Returns the receiver's view.
- [setFullScreen](#) (page 16)  
Sets the receiver to full-screen mode.
- [setOffScreen:width:height:rowbytes:](#) (page 16)  
Instructs the receiver to render into an offscreen buffer with the specified attributes.
- [clearDrawable](#) (page 9)  
Disassociates the receiver from its viewport.
- [update](#) (page 20)  
Updates the receiver's drawable object.

## Flushing the Drawing Buffer

- [flushBuffer](#) (page 11)  
Copies the back buffer to the front buffer of the receiver.

## Copying Attributes

- [copyAttributesFromContext:withMask:](#) (page 9)  
Copies selected groups of state variables to the receiver.

## Context Parameter Handling

- [setValues:forParameter:](#) (page 19)  
Sets the value of the specified parameter.
- [getValues:forParameter:](#) (page 12)  
Returns the value of the requested parameter.

## Working with Virtual Screens

- [setCurrentVirtualScreen:](#) (page 15)  
Sets the current virtual screen for the receiver.
- [currentVirtualScreen](#) (page 10)  
Returns the current virtual screen for the receiver.

## Creating Textures

- [createTexture:fromView:internalFormat:](#) (page 10)  
Creates a new texture from the contents of the specified view.

## Getting the CGL Context Object

- [CGLContextObj](#) (page 8)  
Returns the low-level, platform-specific Core OpenGL (CGL) context object represented by the receiver.

## Working with Pixel Buffers

- [setPixelFormat:cubeMapFace:mipMapLevel:currentVirtualScreen:](#) (page 17)  
Attaches the specified pixel buffer to the receiver.
- [pixelBuffer](#) (page 14)  
Returns the pixel-buffer object attached to the receiver.
- [pixelBufferCubeMapFace](#) (page 14)  
Returns the cube map face of the pixel buffer attached to the receiver.
- [pixelBufferMipMapLevel](#) (page 15)  
Returns the mipmap level of the pixel buffer attached to the receiver.
- [setTextureImageToPixelFormat:colorBuffer:](#) (page 18)  
Attaches the image data in the specified pixel buffer to the texture object currently bound by the receiver.

## Class Methods

### clearCurrentContext

Sets the current context to `nil`.

```
+ (void)clearCurrentContext
```

#### Discussion

Until you issue a subsequent call to the [makeCurrentContext](#) (page 13) method, OpenGL calls do nothing.

#### Availability

Available in Mac OS X v10.0 and later.

**See Also**+ [currentContext](#) (page 8)**Related Sample Code**

GLChildWindowDemo

GLUT

NSOpenGL Fullscreen

OpenGLScreenSnapshot

**Declared In**

NSOpenGL.h

**currentContext**

Returns the current OpenGL graphics context.

+ (NSOpenGLContext \*)currentContext

**Return Value**The current OpenGL graphics context, or `nil` if no such object has been set.**Availability**

Available in Mac OS X v10.0 and later.

**See Also**+ [clearCurrentContext](#) (page 7)+ [currentContext](#) (page 8)- [makeCurrentContext](#) (page 13)**Related Sample Code**

VBL

**Declared In**

NSOpenGL.h

## Instance Methods

**CGLContextObj**

Returns the low-level, platform-specific Core OpenGL (CGL) context object represented by the receiver.

- (void \*)CGLContextObj

**Return Value**A pointer to the `CGLContextObj` data type represented by the receiver.**Availability**

Available in Mac OS X v10.3 and later.

**Related Sample Code**

[Denoise](#)  
[DispatchFractal](#)  
[OpenGLCaptureToMovie](#)  
[Quartz Composer Live DV](#)  
[Quartz Composer Texture](#)

**Declared In**

NSOpenGL.h

**clearDrawable**

Disassociates the receiver from its viewport.

- (void)clearDrawable

**Discussion**

This method disassociates the receiver from any associated `NSView` object. If the receiver is in full-screen or offscreen mode, it exits that mode.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setFullScreen](#) (page 16)
- [setOffScreen:width:height:rowbytes:](#) (page 16)
- [setView:](#) (page 19)
- [view](#) (page 20)

**Related Sample Code**

[GLUT](#)  
[NSOpenGL Fullscreen](#)

**Declared In**

NSOpenGL.h

**copyAttributesFromContext:withMask:**

Copies selected groups of state variables to the receiver.

```
- (void)copyAttributesFromContext:(NSOpenGLContext *)context
    withMask:(GLbitfield)mask
```

**Parameters**

*context*

The OpenGL graphics context containing the desired state variables.

*mask*

A bitfield containing a bitwise OR of the same symbolic names that are passed to the OpenGL call `glPushAttrib`. The single symbolic constant `GL_ALL_ATTRIB_BITS` can be used to copy the maximum possible portion of the rendering state.

**Discussion**

Not all values for OpenGL states can be copied. For example, the pixel pack and unpack state, render mode state, and select and feedback state are not copied. The state that can be copied is exactly the state that is manipulated by the OpenGL call `glPushAttrib`.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSOpenGL.h

**createTexture:fromView:internalFormat:**

Creates a new texture from the contents of the specified view.

```
- (void)createTexture:(GLenum)target fromView:(NSView *)view
    internalFormat:(GLenum)format
```

**Parameters**

*target*

The identifier for the new texture.

*view*

The view to use to generate the texture. This parameter must be either an `NSOpenGLView` object or some other kind of `NSView` object that's associated with an `NSOpenGLContext` object.

*format*

The format for the texture, interpreted as a `GLenum` data type.

**Discussion**

The new texture is assigned the identifier in the *target* parameter and is associated with the receiver's context.

**Availability**

Available in Mac OS X v10.2 and later.

**Related Sample Code**

GLUT

**Declared In**

NSOpenGL.h

**currentVirtualScreen**

Returns the current virtual screen for the receiver.

```
- (GLint)currentVirtualScreen
```

**Return Value**

The virtual screen number, which is a value between 0 and the number of virtual screens minus one.

**Availability**

Available in Mac OS X v10.2 and later.

**See Also**

- [setCurrentVirtualScreen:](#) (page 15)

**Related Sample Code**

Cocoa OpenGL

OpenGL Screensaver

Quartz Composer Offline Rendering

Quartz Composer Texture

**Declared In**

NSOpenGL.h

**flushBuffer**

Copies the back buffer to the front buffer of the receiver.

```
- (void)flushBuffer
```

**Discussion**

If the receiver is not a double-buffered context, this call does nothing.

If the `NSOpenGLPixelFormat` object used to create the context had a `NO` backing store attribute (`NSOpenGLPFABackingStore`), the buffers may be exchanged rather than copied. This is often the case in full-screen mode.

According to the swap interval context attribute (see [NSOpenGLCPSwapInterval](#) (page 21)), the copy may take place during the vertical retrace of the monitor, rather than immediately after `flushBuffer` is called. An implicit `glFlush` is done by `flushBuffer` before it returns. For optimal performance, an application should not call `glFlush` immediately before calling `flushBuffer`. Subsequent OpenGL commands can be issued immediately after calling `flushBuffer`, but are not executed until the buffer copy is completed.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [getValues:forParameter:](#) (page 12)

- [initWithFormat:shareContext:](#) (page 12)

- [setValues:forParameter:](#) (page 19)

**Related Sample Code**

DispatchLife

NSOpenGL Fullscreen

NURBSSurfaceVertexProg

SurfaceVertexProgram

VideoViewer

**Declared In**

NSOpenGL.h

**getValues:forParameter:**

Returns the value of the requested parameter.

```
- (void)getValues:(GLint *)vals forParameter:(NSOpenGLContextParameter)param
```

**Parameters**

*vals*

On input, a pointer to a variable with enough space for one or more long integers. On output, the variable contains the value (or values) for the given parameter.

*param*

The parameter you want to get. For a list of parameters, see the table in [NSOpenGLContextParameter](#) (page 21).

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setValues:forParameter:](#) (page 19)

**Declared In**

NSOpenGL.h

**initWithCGLContextObj:**

Initializes and returns a `NSOpenGLContext` object using an existing CGL context.

```
- (id)initWithCGLContextObj:(void *)context
```

**Parameters**

*context*

The CGL context to wrap inside the `NSOpenGLContext` object.

**Return Value**

An initialized context.

**Discussion**

If your application already has a CGL context, you can wrap a `NSOpenGLContext` object around it using this method. This method retains the CGL context by calling `CGLRetainContext`.

Only one `NSOpenGLContext` object can wrap a specific context.

Your application should not call `CGLDestroyContext` to dispose of the CGL context. Instead, your application should call `CGLReleaseContext` to decrement its reference count.

**Availability**

Available in Mac OS X v10.6 and later.

**Declared In**

NSOpenGL.h

**initWithFormat:shareContext:**

Returns an `NSOpenGLContext` object initialized with the specified pixel format information.

```
- (id)initWithFormat:(NSOpenGLPixelFormat *)format shareContext:(NSOpenGLContext *)share
```

### Parameters

*format*

The pixel format to request for the OpenGL graphics context. Following successful initialization, the value you pass in for this parameter is no longer needed and can be deallocated.

*share*

Another OpenGL graphics context whose texture namespace and display lists you want to share with the receiver. If you do not want to share those features with another graphics context, you may pass `nil` for this parameter.

### Return Value

An `NSOpenGLContext` object initialized with the specified parameters, or `nil` if the object could not be created.

### Discussion

If the parameters contain invalid information, the receiver releases itself and this method returns `nil`. This may happen if one of the following situations occurs:

- The *format* parameter is `nil` or contains an invalid pixel format.
- The *share* parameter is not `nil` and contains an invalid context.
- The *share* parameter contains a context with a pixel format that is incompatible with the one in *format*.

Pixel formats are incompatible if they use different renderers; this can happen if, for example, one format required an accumulation buffer that could only be provided by the software renderer, and the other format did not.

### Availability

Available in Mac OS X v10.0 and later.

### Related Sample Code

CubePuzzle

GLUT

LiveVideoMixer2

NSOpenGL Fullscreen

Quartz Composer Texture

### Declared In

NSOpenGL.h

## makeCurrentContext

Sets the receiver as the current OpenGL context object.

```
- (void)makeCurrentContext
```

### Discussion

Subsequent OpenGL calls are rendered into the context defined by the receiver.

**Note:** A context is current on a per-thread basis. Multiple threads must serialize calls into the same context object.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

+ [clearCurrentContext](#) (page 7)

+ [currentContext](#) (page 8)

**Related Sample Code**

GLUT

LiveVideoMixer2

NURBSSurfaceVertexProg

OpenGL Filter Basics Cocoa

SurfaceVertexProgram

**Declared In**

NSOpenGL.h

**pixelBuffer**

Returns the pixel-buffer object attached to the receiver.

- (NSOpenGLPixelFormat \*)pixelBuffer

**Return Value**

The pixel buffer object.

**Availability**

Available in Mac OS X v10.3 and later.

**See Also**

- [setPixelFormat:cubeMapFace:mipMapLevel:currentVirtualScreen:](#) (page 17)

**Declared In**

NSOpenGL.h

**pixelBufferCubeMapFace**

Returns the cube map face of the pixel buffer attached to the receiver.

- (GLenum)pixelBufferCubeMapFace

**Return Value**

For pixel buffers with a texture target of `GL_CUBE_MAP`, this value is zero or one of the following values:

- `GL_TEXTURE_CUBE_MAP_POSITIVE_X`
- `GL_TEXTURE_CUBE_MAP_POSITIVE_Y`
- `GL_TEXTURE_CUBE_MAP_POSITIVE_Z`

- `GL_TEXTURE_CUBE_MAP_NEGATIVE_X`
- `GL_TEXTURE_CUBE_MAP_NEGATIVE_Y`
- `GL_TEXTURE_CUBE_MAP_NEGATIVE_Z`

**Availability**

Available in Mac OS X v10.3 and later.

**See Also**

- [setPixelFormat:cubeMapFace:mipMapLevel:currentVirtualScreen:](#) (page 17)

**Declared In**

NSOpenGL.h

**pixelBufferMipMapLevel**

Returns the mipmap level of the pixel buffer attached to the receiver.

- (GLint)pixelBufferMipMapLevel

**Return Value**

The desired mipmap level for rendering. This value should be less than or equal to the maximum texture mipmap level of *pixelBuffer* (accessible through an `NSOpenGLPixelFormat` object's `textureMaxMipMapLevel` method).

**Availability**

Available in Mac OS X v10.3 and later.

**See Also**

- [setPixelFormat:cubeMapFace:mipMapLevel:currentVirtualScreen:](#) (page 17)

**Declared In**

NSOpenGL.h

**setCurrentVirtualScreen:**

Sets the current virtual screen for the receiver.

- (void)setCurrentVirtualScreen:(GLint)*screen*

**Parameters**

*screen*

The virtual screen number, which is a value between 0 and the number of virtual screens minus one.

**Availability**

Available in Mac OS X v10.2 and later.

**See Also**

- [currentVirtualScreen](#) (page 10)

**Declared In**

NSOpenGL.h

## setFullScreen

Sets the receiver to full-screen mode.

```
- (void)setFullScreen
```

### Discussion

In full-screen mode, the receiver renders onto the entire screen. The receiver's viewport is set to the full size of the screen. Call the `clearDrawable` (page 9) method to exit full-screen mode.

The `NSOpenGLPFAFullScreen` attribute must have been specified in the receiver's `NSOpenGLPixelFormat`. Some OpenGL renderers, like the software renderer, do not support full-screen mode. The following code determines if a full-screen pixel format is possible on a given system:

```
NSOpenGLPixelFormatAttribute attrs[] =
{
    NSOpenGLPFAFullScreen,
    nil
};

NSOpenGLPixelFormat* pixFmt = [[NSOpenGLPixelFormat alloc]
initWithAttributes:attrs];

/* Check if initWithAttributes succeeded. */
if(pixFmt == nil) {
    /* initWithAttributes failed. There is no full-screen renderer. */
}
```

**Note:** It is recommended that an application use Core Graphics's **Direct Display** API to capture the display before entering full-screen mode and release it after exiting. A captured display prevents contention from other applications and system services. In addition, applications are not notified of display changes, preventing them from repositioning their windows and the Finder from repositioning desktop icons.

### Availability

Available in Mac OS X v10.0 and later.

### Related Sample Code

DispatchLife

GLUT

NSOpenGL Fullscreen

VBL

### Declared In

NSOpenGL.h

## setOffScreen:width:height:rowbytes:

Instructs the receiver to render into an offscreen buffer with the specified attributes.

```
- (void)setOffScreen:(void *)baseaddr width:(GLsizei)width height:(GLsizei)height
rowbytes:(GLint)rowbytes
```

**Parameters***baseaddr*

The base address of the buffer in memory. This buffer must contain at least  $rowbytes * height$  bytes.

*width*

The width of the memory buffer, measured in pixels.

*height*

The height of the memory buffer, measured in pixels.

*rowbytes*

The number of bytes in a single row of the buffer. This value must be greater than or equal to the value in *width* times the number of bytes per pixel.

**Discussion**

The receiver's viewport is set to the full size of the offscreen area. Call the `clearDrawable` (page 9) method to exit offscreen mode.

The `NSOpenGLPFAOffScreen` attribute must have been specified in the receiver's pixel format object.

**Note:** To obtain behavior similar to offscreen mode on renderers that do not support accelerated offscreen contexts, attach the context to a hidden window and use `glReadPixels`.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSOpenGL.h

**setPixelBuffer:cubeMapFace:mipMapLevel:currentVirtualScreen:**

Attaches the specified pixel buffer to the receiver.

```
- (void)setPixelBuffer:(NSOpenGLPixelBuffer *)pixelBuffer cubeMapFace:(GLenum)face
      mipMapLevel:(GLint)level currentVirtualScreen:(GLint)screen
```

**Parameters***pixelBuffer*

The pixel buffer to attach.

*face*

For pixel buffers with a texture target of `GL_CUBE_MAP`, this parameter should be zero or one of the following values:

- `GL_TEXTURE_CUBE_MAP_POSITIVE_X`
- `GL_TEXTURE_CUBE_MAP_POSITIVE_Y`
- `GL_TEXTURE_CUBE_MAP_POSITIVE_Z`
- `GL_TEXTURE_CUBE_MAP_NEGATIVE_X`
- `GL_TEXTURE_CUBE_MAP_NEGATIVE_Y`
- `GL_TEXTURE_CUBE_MAP_NEGATIVE_Z`

*level*

The desired mipmap level for rendering. This value must be less than or equal to the maximum texture mipmap level of *pixelBuffer* (accessible through an `NSOpenGLPixelBuffer` object's `textureMaxMipMapLevel` method).

*screen*

The virtual screen of the receiver (if applicable) should be set to the same value as the current virtual screen you are using for rendering onscreen.

**Discussion**

The `NSOpenGLPixelBuffer` object gives the receiver access to accelerated offscreen rendering in the pixel buffer, which is primarily used for textures.

**Availability**

Available in Mac OS X v10.3 and later.

**See Also**

- [pixelBufferCubeMapFace](#) (page 14)
- [pixelBufferMipMapLevel](#) (page 15)
- [setCurrentVirtualScreen:](#) (page 15)
- `initWithTextureTarget:textureInternalFormat:textureMaxMipMapLevel:pixelsWide:pixelsHigh:` (`NSOpenGLPixelBuffer`)

**Related Sample Code**

Quartz Composer Offline Rendering

Quartz Composer Texture

**Declared In**

`NSOpenGL.h`

**setTextureImageToPixelBuffer:colorBuffer:**

Attaches the image data in the specified pixel buffer to the texture object currently bound by the receiver.

```
(void)setTextureImageToPixelBuffer:(NSOpenGLPixelBuffer *)pixelBuffer
    colorBuffer:(GLenum)source
```

**Parameters**

*pixelBuffer*

The pixel buffer to attach.

*source*

An OpenGL constant indicating which of the pixel buffer's color buffers to use. Potential values for this parameter include `GL_FRONT`, `GL_BACK`, and `GL_AUX0`.

**Discussion**

This method corresponds to the Core OpenGL method `CGLTexImagePBuffer`.

**Availability**

Available in Mac OS X v10.3 and later.

**Related Sample Code**

Quartz Composer Texture

**Declared In**  
NSOpenGL.h

## setValues:forParameter:

Sets the value of the specified parameter.

```
- (void)setValues:(const GLint *)vals forParameter:(NSOpenGLContextParameter)param
```

### Parameters

*vals*

The new value (or values) for the parameter.

*param*

The parameter you want to modify. For a list of parameters, see [NSOpenGLContextParameter](#) (page 21).

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [getValues:forParameter:](#) (page 12)

### Related Sample Code

Denoise

GLSLShowpiece

GLUT

LiveVideoMixer3

QTCoreVideo201

**Declared In**  
NSOpenGL.h

## setView:

Sets the receiver's viewport to the specified `NSView` object.

```
- (void)setView:(NSView *)view
```

### Parameters

*view*

The view to use for drawing. The full size of the view is used for the viewport.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [clearDrawable](#) (page 9)

- [view](#) (page 20)

### Related Sample Code

GLUT

LiveVideoMixer

LiveVideoMixer2

LiveVideoMixer3

**Declared In**

NSOpenGL.h

**update**

Updates the receiver's drawable object.

- (void)update

**Discussion**

Call this method whenever the receiver's drawable object changes size or location. A multithreaded application must synchronize all threads that access the same drawable object and call `update` for each thread's context serially.

**Availability**

Available in Mac OS X v10.0 and later.

**Related Sample Code**

DispatchLife

Quartz Composer Live DV

TexturePerformanceDemo

TextureRange

VideoViewer

**Declared In**

NSOpenGL.h

**view**

Returns the receiver's view.

- (NSView \*)view

**Return Value**The view, or `nil` if the receiver has no drawable object, is in full-screen mode, or is in offscreen mode.**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [clearDrawable](#) (page 9)
- [setFullScreen](#) (page 16)
- [setOffScreen:width:height:rowbytes:](#) (page 16)
- [setView:](#) (page 19)

**Declared In**

NSOpenGL.h

## Constants

### NSOpenGLContextParameter

The following attribute names are used by [setValues:forParameter:](#) (page 19) and [getValues:forParameter:](#) (page 12):

```
typedef enum {
    NSOpenGLCPSwapRectangle = 200,
    NSOpenGLCPSwapRectangleEnable = 201,
    NSOpenGLCPRasterizationEnable = 221,
    NSOpenGLCPSwapInterval = 222,
    NSOpenGLCPSurfaceOrder = 235,
    NSOpenGLCPSurfaceOpacity = 236,
    NSOpenGLCPStateValidation = 301
} NSOpenGLContextParameter;
```

#### Constants

NSOpenGLCPSwapRectangle

Sets or gets the swap rectangle.

The swap rectangle is represented as an array of four longs: {x, y, width, height}.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

NSOpenGLCPSwapRectangleEnable

Enables or disables the swap rectangle in the context's drawable object.

If enabled, the area that is affected by the [flushBuffer](#) (page 11) method is restricted to a rectangle specified by the values of `NSOpenGLCPSwapRectangle`. However, the portion of the drawable object that lies outside of the swap rectangle may still be flushed to the screen by a visibility change or other user interface action.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

NSOpenGLCPRasterizationEnable

If disabled, all rasterization of 2D and 3D primitives is disabled.

This state is useful for debugging and to characterize the performance of an OpenGL driver without actually rendering.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

NSOpenGLCPSwapInterval

Sets or gets the swap interval.

The swap interval is represented as one long. If the swap interval is set to 0 (the default), the [flushBuffer](#) (page 11) method executes as soon as possible, without regard to the vertical refresh rate of the monitor. If the swap interval is set to 1, the buffers are swapped only during the vertical retrace of the monitor.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

NSOpenGLCPSurfaceOrder

Get or set the surface order.

If the surface order is set to 1 (the default), the order is above the window (default). If the value is -1, the order is below the window.

Available in Mac OS X v10.2 and later.

Declared in `NSOpenGL.h`.

NSOpenGLCPSurfaceOpacity

Set or get the surface opacity.

If the opacity is set to 1 (the default), the surface is opaque. If the value is 0, the surface is non-opaque.

Available in Mac OS X v10.2 and later.

Declared in `NSOpenGL.h`.

NSOpenGLCPStateValidation

If enabled, OpenGL inspects the context state each time the [update](#) (page 20) method is called to ensure that it is in an appropriate state for switching between renderers.

Normally, the state is inspected only when it is actually necessary to switch renderers. This is useful when using a single monitor system to test that an application performs correctly on a multiple-monitor system.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`NSOpenGL.h`

# Document Revision History

---

This table describes the changes to *NSOpenGLContext Class Reference*.

Date	Notes
2009-08-13	Added Mac OS X v10.6 symbols.
2008-06-09	Corrected description of <code>NSOpenGLCPSwapInterval</code> .
2007-01-26	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

## REVISION HISTORY

### Document Revision History

# Index

---

## C

---

CGLContextObj **instance method** [8](#)  
clearCurrentContext **class method** [7](#)  
clearDrawable **instance method** [9](#)  
copyAttributesFromContext:withMask: **instance method** [9](#)  
createTexture:fromView:internalFormat: **instance method** [10](#)  
currentContext **class method** [8](#)  
currentVirtualScreen **instance method** [10](#)

## F

---

flushBuffer **instance method** [11](#)

## G

---

getValues:forParameter: **instance method** [12](#)

## I

---

initWithCGLContextObj: **instance method** [12](#)  
initWithFormat:shareContext: **instance method** [12](#)

## M

---

makeCurrentContext **instance method** [13](#)

## N

---

NSOpenGLContextParameter **data type** [21](#)  
NSOpenGLCPRasterizationEnable **constant** [21](#)

NSOpenGLCPStateValidation **constant** [22](#)  
NSOpenGLCPSurfaceOpacity **constant** [22](#)  
NSOpenGLCPSurfaceOrder **constant** [22](#)  
NSOpenGLCPSwapInterval **constant** [21](#)  
NSOpenGLCPSwapRectangle **constant** [21](#)  
NSOpenGLCPSwapRectangleEnable **constant** [21](#)

## P

---

pixelBuffer **instance method** [14](#)  
pixelBufferCubeMapFace **instance method** [14](#)  
pixelBufferMipMapLevel **instance method** [15](#)

## S

---

setCurrentVirtualScreen: **instance method** [15](#)  
setFullScreen **instance method** [16](#)  
setOffScreen:width:height:rowbytes: **instance method** [16](#)  
setPixelBuffer:cubeMapFace:mipMapLevel:currentVirtualScreen: **instance method** [17](#)  
setTextureImageToPixelBuffer:colorBuffer: **instance method** [18](#)  
setValues:forParameter: **instance method** [19](#)  
setView: **instance method** [19](#)

## U

---

update **instance method** [20](#)

## V

---

view **instance method** [20](#)