NSOpenGLContext Class Reference

Cocoa > Graphics & Imaging

2008-06-09

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

Contents

NSOpenGLContext Class Reference

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

Tasks

Context Creation

`- initWithFormat:shareContext:` (page 11)
Returns an `NSOpenGLContext` object initialized with the specified pixel format information.

Managing the Current Context

`+ clearCurrentContext` (page 7)
Sets the current context to `nil`.

+ `currentContext` (page 8)
Returns the current OpenGL graphics context.

- `makeCurrentContext` (page 12)
Sets the receiver as the current OpenGL context object.

Drawable Object Management

- `setView:` (page 18)
Sets the receiver's viewport to the specified `NSView` object.

- `view` (page 19)
Returns the receiver's view.

- `setFullScreen` (page 15)
Sets the receiver to full-screen mode.

- `setOffScreen:width:height:rowbytes:` (page 15)
Instructs the receiver to render into an offscreen buffer with the specified attributes.

- `clearDrawable` (page 9)
Disassociates the receiver from its viewport.

- `update` (page 18)
Updates the receiver's drawable object.

Flushing the Drawing Buffer

- `flushBuffer` (page 10)
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 17)
Sets the value of the specified parameter.

- `getValues:forParameter:` (page 11)
Returns the value of the requested parameter.

Working with Virtual Screens

- `setCurrentVirtualScreen:` (page 14)
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

- `setPixelBuffer:cubeMapFace:mipMapLevel:currentVirtualScreen:` (page 16)
  Attaches the specified pixel buffer to the receiver.
- `pixelBuffer` (page 13)
  Returns the pixel-buffer object attached to the receiver.
- `pixelBufferCubeMapFace` (page 13)
  Returns the cube map face of the pixel buffer attached to the receiver.
- `pixelBufferMipMapLevel` (page 14)
  Returns the mipmap level of the pixel buffer attached to the receiver.
- `setTextureImageToPixelBuffer:colorBuffer:` (page 17)
  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 12) method, OpenGL calls do nothing.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ currentContext` (page 8)

Related Sample Code
GLChildWindowDemo
NSOpenGL Fullscreen

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 12)

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
OpenGLCaptureToMovie
Quartz Composer Live DV
Quartz Composer QCTV
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 15)
- setOffScreen:width:height:rowbytes: (page 15)
- setView: (page 18)
- view (page 19)

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.

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 14)

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 20)), 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 11)
- `- initWithFormat:shareContext:` (page 11)
- `- setValues:forParameter:` (page 17)

Related Sample Code
VBL

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 19).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- setValues:forParameter:` (page 17)

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
GLSLShowpiece
LiveVideoMixer3
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
GLSLShowpiece
LiveVideoMixer2
VBL

Declared In
`NSOpenGL.h`

pixelBuffer

Returns the pixel-buffer object attached to the receiver.

```
- (NSOpenGLPixelBuffer *)pixelBuffer
```

Return Value
The pixel buffer object.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setPixelBuffer:cubeMapFace:mipMapLevel:currentVirtualScreen:` (page 16)

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
- setPixelBuffer:cubeMapFace:mipMapLevel:currentVirtualScreen: (page 16)

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 NSOpenGLPixelBuffer object's textureMaxMipMapLevel method).

Availability
Available in Mac OS X v10.3 and later.

See Also
- setPixelBuffer:cubeMapFace:mipMapLevel:currentVirtualScreen: (page 16)

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
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 13)
- `pixelBufferMipMapLevel` (page 14)
- `setCurrentVirtualScreen:` (page 14)
- `initWithTextureTarget:textureInternalFormat:textureMaxMipMapLevel:pixelsWide:pixelsHigh:` (`NSOpenGLPixelBuffer`)

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.

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 19).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `getValues:forParameter:` (page 11)

Related Sample Code
LiveVideoMixer2

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 19)

Related Sample Code
LiveVideoMixer2

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.

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 15)
- `setOffScreen:width:height:rowbytes:` (page 15)
- `setView:` (page 18)

Declared In
`NSOpenGL.h`

Constants

NSOpenGLContextParameter

The following attribute names are used by `setValues:forParameter:` (page 17) and `getValues:forParameter:` (page 11):

```
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 `long`s: `{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 10) 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 10) 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 18) 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
2008-06-09	Corrected description of NSOpenGLCPSwapInterval.
2007-01-26	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

Document Revision History

Index