AGL Reference

Graphics & Imaging > OpenGL

2007-10-31

Apple Inc.
© 2004, 2007 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, Carbon, Mac, Mac OS, Macintosh, and Tiger are trademarks of Apple Inc., registered in the United States and other countries.

Intel and Intel Core are registered trademarks of Intel Corportation or its subsidiaries in the United States and other countries.

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

AGL Reference 7

Overview 7
Functions by Task 7
- Managing Pixel Format Objects 7
- Managing Contexts 8
- Getting and Setting Context Options 8
- Managing Drawable Objects 8
- Managing Pixel Buffers 9
- Using a Window as a Texture Source 9
- Getting Error Information 9
- Getting and Setting Global Information 9
- Getting Renderer Information 10
- Generating Bitmap Display Lists 10
- Managing Virtual Screens 10
- Getting and Setting Windows 10
- Getting and Setting HIView Objects 10

Functions 11
- aglChoosePixelFormat 11
- aglConfigure 13
- aglCopyContext 13
- aglCreateContext 14
- aglCreatePBuffer 15
- aglCreatePixelFormat 16
- aglDescribePBuffer 17
- aglDescribePixelFormat 18
- aglDescribeRenderer 18
- aglDestroyContext 19
- aglDestroyPBuffer 20
- aglDestroyPixelFormat 21
- aglDestroyRendererInfo 21
- aglDevicesOfPixelFormat 22
- aglDisable 23
- aglDisplaysOfPixelFormat 23
- aglEnable 24
- aglErrorString 25
- aglGetCGLContext 25
- aglGetCGLPixelFormat 26
- aglGetCurrentContext 27
- aglGetDrawable 27
- aglGetError 28
- aglGetHIViewRef 28

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

aglGetInteger 29
aglGetPBuffer 29
aglGetVersion 30
aglGetVirtualScreen 31
aglGetWindowRef 31
aglIsEnabled 32
aglNextPixelFormat 32
aglNextRendererInfo 33
aglQueryRendererInfo 34
aglQueryRendererInfoForCGDirectDisplayIDs 35
aglResetLibrary 35
aglSetCurrentContext 35
aglSetDrawable 36
aglSetFullScreen 37
aglSetHIViewRef 38
aglSetInteger 38
aglSetOffScreen 39
aglSetPBuffer 40
aglSetVirtualScreen 41
aglSetWindowRef 42
aglSurfaceTexture 43
aglSwapBuffers 44
aglTexImagePBuffer 44
aglUpdateContext 46
aglUseFont 46
Data Types 48
AGLContext 48
AGLDevice 48
AGLDrawable 48
AGLPbuffer 49
AGLPixelFormat 49
AGLRendererInfo 50
Constants 50
Bit Depths 50
Buffer and Renderer Attributes 52
Buffer Mode Flags 57
Color Modes 58
Context Options and Parameters 63
Error Codes 66
Globally Configured Options 69
Renderer Attributes 70
Renderer IDs 72
Renderer Properties 74

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Document Revision History 77

Index 79

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

AGL Reference

Framework:	AGL/agl.h
Companion guide	OpenGL Programming Guide for Mac OS X
Declared in	agl.h aglRenderers.h

Overview

The AGL (Apple Graphics Library) API is part of the Apple implementation of OpenGL in Mac OS X. AGL contains the windowing system–specific implementation of OpenGL functions and commands for Carbon. Using AGL functions, you can create and destroy AGL rendering contexts, select OpenGL renderers, swap buffers, and perform operations on drawable objects (windows, pixel buffers, offscreen memory, and full-screen graphics devices).

To use AGL from the Carbon framework, your application must link to both the AGL and the OpenGL (`OpenGL/OpenGL.h`) frameworks.

Note: This document was previously titled *Apple OpenGL Reference*.

Functions by Task

Managing Pixel Format Objects

`aglCreatePixelFormat` (page 16)
Creates a pixel format with the provided attributes.

`aglChoosePixelFormat` (page 11)
Creates a pixel format object that satisfies the constraints of the specified buffer and renderer attributes.

`aglDescribePixelFormat` (page 18)
Retrieves the value of an attribute associated with a pixel format object.

`aglDestroyPixelFormat` (page 21)
Frees the memory associated with a pixel format object.

`aglGetCGLPixelFormat` (page 26)
Gets the CGL pixel format object associated with an AGL pixel format.

`aglDisplaysOfPixelFormat` (page 23)
Returns the graphics devices supported by a pixel format object.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`aglDevicesOfPixelFormat` (page 22)
Returns the graphics devices supported by a pixel format object. (Deprecated. Use `aglDisplaysOfPixelFormat` (page 23) instead.)

`aglNextPixelFormat` (page 32)
Returns the next pixel format object in a list of pixel format objects.

Managing Contexts

`aglCreateContext` (page 14)
Creates an AGL rendering context.

`aglCopyContext` (page 13)
Copies the specified state variables from one rendering context to another.

`aglDestroyContext` (page 19)
Frees the resources associated with a rendering context.

`aglGetCGLContext` (page 25)
Gets the CGL rendering context associated with an AGL rendering context.

`aglGetCurrentContext` (page 27)
Returns the current rendering context.

`aglSetCurrentContext` (page 35)
Sets the specified rendering context as the current rendering context.

`aglSwapBuffers` (page 44)
Exchanges the front and back buffers of the specified rendering context.

`aglUpdateContext` (page 46)
Notifies the rendering context that the window geometry has changed.

Getting and Setting Context Options

`aglEnable` (page 24)
Enables an option for a rendering context.

`aglDisable` (page 23)
Disables an option for a rendering context.

`aglIsEnabled` (page 32)
Reports whether an option is enabled for a rendering context.

`aglSetInteger` (page 38)
Sets the value of an option for a rendering context.

`aglGetInteger` (page 29)
Retrieves the integer setting of an AGL context option.

Managing Drawable Objects

`aglGetDrawable` (page 27)
Returns the drawable object attached to a rendering context.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`aglSetDrawable` (page 36)
Attaches a rendering context to a Carbon window.

`aglSetFullScreen` (page 37)
Attaches a rendering context to a full screen graphics device.

`aglSetOffScreen` (page 39)
Attaches a rendering context to an offscreen memory area.

Managing Pixel Buffers

`aglCreatePBuffer` (page 15)
Creates a pixel buffer of the specified size, compatible with the specified texture target.

`aglDestroyPBuffer` (page 20)
Releases the resources associated with a pixel buffer object.

`aglDescribePBuffer` (page 17)
Retrieves information that describes the specified pixel buffer object.

`aglGetPBuffer` (page 29)
Retrieves a pixel buffer and its parameters for a specified rendering context.

`aglSetPBuffer` (page 40)
Attaches a pixel buffer object to a rendering context.

`aglTexImagePBuffer` (page 44)
Binds the contents of a pixel buffer to a data source for a texture object.

Using a Window as a Texture Source

`aglSurfaceTexture` (page 43)
Allows texturing from a drawable object that has an attached rendering context, using the surface contents as the source data for the texture.

Getting Error Information

`aglGetError` (page 28)
Returns an AGL error code.

`aglErrorString` (page 25)
Returns a string that describes the specified AGL error code.

Getting and Setting Global Information

`aglConfigure` (page 13)
Sets the value of a global option.

`aglGetVersion` (page 30)
Gets the major and minor version numbers of the AGL library.

`aglResetLibrary` (page 35)
Resets the AGL library. (Deprecated. This function is not needed in Mac OS X.)

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Getting Renderer Information

`aglDescribeRenderer` (page 18)
Obtains the value associated with a renderer property.

`aglDestroyRendererInfo` (page 21)
Frees resources associated with a renderer information object.

`aglNextRendererInfo` (page 33)
Returns the next renderer information object.

`aglQueryRendererInfo` (page 34)
Creates and returns a renderer information object that contains properties and values for all renderers driving the specified displays. (Deprecated. Instead use `aglQueryRendererInfoForCGDirectDisplayIDs` (page 35).)

`aglQueryRendererInfoForCGDirectDisplayIDs` (page 35)
Creates and returns a renderer information object that contains properties and values for all renderers driving the specified displays.

Generating Bitmap Display Lists

`aglUseFont` (page 46)
Creates bitmap display lists from a font.

Managing Virtual Screens

`aglGetVirtualScreen` (page 31)
Returns the current virtual screen number associated with a rendering context.

`aglSetVirtualScreen` (page 41)
Forces subsequent OpenGL commands to the specified virtual screen.

Getting and Setting Windows

`aglSetWindowRef` (page 42)
Sets an AGL context to the specified window.

`aglGetWindowRef` (page 31)
Retrieves the window associated with an AGL context.

Getting and Setting HIView Objects

`aglSetHIViewRef` (page 38)
Sets an AGL context to the specified HIView object.

`aglGetHIViewRef` (page 28)
Retrieves the HIView object associated with an AGL context.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Functions

aglChoosePixelFormat

Creates a pixel format object that satisfies the constraints of the specified buffer and renderer attributes.

```
AGLPixelFormat aglChoosePixelFormat (
   const void *gdevs,
   GLint ndev,
   const GLint *attribs
);
```

Parameters

gdevs

A pointer to an AGLDevice (page 48) data type that contains an array of Mac OS graphics devices. AGL chooses pixel formats and renderers that are appropriate for these devices. To create a pixel format object that supports all devices on the system, pass NULL.

ndev

The number of graphics devices that your application supplies in the gdevs parameter. Pass 0 if you also pass NULL for the *gdevs* parameter.

attribs

A NULL terminated array that contains a list of buffer and renderer attributes. Attributes can be Boolean or integer. If an attribute is integer, you must supply the desired value immediately following the attribute. If the attribute is Boolean, do not supply a value because its presence in the attributes array implies a true value. For information on the attributes that you can supply, see "Buffer and Renderer Attributes" (page 52), "Renderer Attributes" (page 70), and the Discussion below.

Return Value

A new pixel format object that contains pixel format information and a list of virtual screens. Returns NULL if the system cannot find a pixel format and virtual screen that satisfies the constraints of the buffer and renderer attributes.

Discussion

After a pixel format object is created successfully, the integer attributes are set to values that are as close to the desired value as can be provided by the system. Attribute values can differ for each virtual screen. You can use the AGL_MINIMUM_POLICY (page 56) and AGL_MAXIMUM_POLICY (page 56) attributes to control how the system chooses the setting. For more information on choosing attributes, see *OpenGL Programming Guide for Mac OS X*.

The Boolean attribute constants include the following. For complete descriptions, see "Buffer and Renderer Attributes" (page 52) and "Renderer Attributes" (page 70).

AGL_ALL_RENDERERS (page 53)
AGL_RGBA (page 54)
AGL_DOUBLEBUFFER (page 54)
AGL_STEREO (page 54)
AGL_MINIMUM_POLICY (page 56)
AGL_MAXIMUM_POLICY (page 56)
AGL_OFFSCREEN (page 56)
AGL_FULLSCREEN (page 56)
AGL_COLOR_FLOAT (page 57)

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

AGL_MULTISAMPLE (page 57)
AGL_SUPERSAMPLE (page 57)
AGL_SAMPLE_ALPHA (page 57)
AGL_SINGLE_RENDERER (page 70)
AGL_NO_RECOVERY (page 71)
AGL_ACCELERATED (page 71)
AGL_ROBUST (page 71)
AGL_BACKING_STORE (page 71)
AGL_WINDOW (page 72)
AGL_MULTISCREEN (page 72)
AGL_COMPLIANT (page 72)
AGL_PBUFFER (page 72)
AGL_REMOTE_PBUFFER (page 72)

The following are integer attribute constants and must be followed by a value. For complete descriptions, see "Buffer and Renderer Attributes" (page 52) and "Renderer Attributes" (page 70).

AGL_BUFFER_SIZE (page 53)
AGL_LEVEL (page 54)
AGL_AUX_BUFFERS (page 54)
AGL_RED_SIZE (page 54)
AGL_GREEN_SIZE (page 54)
AGL_BLUE_SIZE (page 55)
AGL_ALPHA_SIZE (page 55)
AGL_DEPTH_SIZE (page 55)
AGL_STENCIL_SIZE (page 55)
AGL_ACCUM_RED_SIZE (page 55)
AGL_ACCUM_GREEN_SIZE (page 55)
AGL_ACCUM_BLUE_SIZE (page 55)
AGL_ACCUM_ALPHA_SIZE (page 55)
AGL_PIXEL_SIZE (page 56)
AGL_SAMPLE_BUFFERS_ARB (page 56)
AGL_SAMPLES_ARB (page 57)
AGL_AUX_DEPTH_STENCIL (page 57)

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

See Also
aglCreateContext (page 14)
aglDescribePixelFormat (page 18)
aglDestroyPixelFormat (page 21)

Related Sample Code
aglClipBufferRect
Carbon GLSnapshot
GLCarbon1ContextPbuffer
GLCarbonSharedPbuffer

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

QTCarbonCoreImage101

Declared In
`agl.h`

aglConfigure

Sets the value of a global option.

```
GLboolean aglConfigure (
   GLenum pname,
   GLuint param
);
```

Parameters

pname

The name of the option whose value you want to set. See "Globally Configured Options" (page 69) for a list of constants you can pass.

param

The value to set the option to.

Return Value
Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Discussion
This function changes the values of options that affect the operation of OpenGL in all rendering contexts in the application, not just the current rendering context.

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

Declared In
`agl.h`

aglCopyContext

Copies the specified state variables from one rendering context to another.

```
GLboolean aglCopyContext (
   AGLContext src,
   AGLContext dst,
   GLuint mask
);
```

Parameters

src

The source rendering context.

dst

The destination rendering context.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

mask

A mask that specifies the state variables to copy. Pass a bit field that contains the bitwise `OR` of the state variable names that you want to copy. Use the symbolic mask constants that are passed to the OpenGL function `glPushAttrib`. To copy as many state variables as possible, supply the constant `GL_ALL_ATTRIB_BITS`. For a description of the symbolic mask constants, see OpenGL Reference Manual.

Return Value
Returns `GL_FALSE` if the function fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Discussion
Not all OpenGL state values can be copied. For example, 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
`agl.h`

aglCreateContext

Creates an AGL rendering context.

```
AGLContext aglCreateContext (
   AGLPixelFormat pix,
   AGLContext share
);
```

Parameters

pix

A pixel format object creating by calling the function aglChoosePixelFormat (page 11).

share

The rendering context with which to share the OpenGL object state—including texture objects, programs and shader display lists, vertex array objects, vertex buffer objects, pixel buffer objects, and frame buffer objects—and the object state associated which each of these object types. Pass `NULL` to indicate that no sharing is to take place.

Return Value
A new rendering context. The `aglCreateContext` function returns `NULL` if the function fails for any reason. You can call the function `aglGetError` (page 28) to determine what the error is.

Discussion
If the pixel format object you supply is able to support multiple graphics devices, then the rendering context can render transparently across the supported devices. With a multiple-device rendering context, sharing is possible only when the relationship between the renderers and the graphics devices they support is the same for all rendering contexts that are shared. Normally you achieve the best results by using the same pixel format object for all shared rendering contexts. For more information, see *OpenGL Programming Guide for Mac OS X*.

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

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

See Also
aglChoosePixelFormat (page 11)
aglDestroyContext (page 19)
aglSetDrawable (page 36)

Related Sample Code
AGLSurfaceTexture
Carbon GLSnapshot
GLCarbon1ContextPbuffer
GLCarbonSharedPbuffer
OpenGLMovieQT

Declared In
agl.h

aglCreatePBuffer

Creates a pixel buffer of the specified size, compatible with the specified texture target.

```
GLboolean aglCreatePBuffer (
   GLint width,
   GLint height,
   GLenum target,
   GLenum internalFormat,
   GLint max_level,
   AGLPbuffer *pbuffer
);
```

Parameters

width
The width, in pixels, of the pixel buffer.

height
The height, in pixels, of the pixel buffer.

target
A constant that specifies the type of the pixel buffer target texture. You can supply any of the following texture targets:

- GL_TEXTURE_2D, a texture whose dimensions are a power of two.
- GL_TEXTURE_RECTANGLE_EXT, a texture whose dimensions are not a power of two.
- GL_TEXTURE_CUBE_MAP, a mapped cube texture.

internalFormat
A constant that specifies the internal color format of the pixel buffer, which can be GL_RGB or GL_RGBA. The format controls whether the alpha channel of the pixel buffer is used for texturing operations.

max_level
The maximum level of mipmap detail allowable. Pass 0 for a pixel buffer that's not using mipmaps. The value passed should never exceed the actual maximum number of mipmap levels that can be represented with the given width and height.

pbuffer
On return, points to a new pixel buffer object.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Return Value
Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Discussion
This function does not have any knowledge of OpenGL contexts or pixel format objects and does not specifically allocate the storage needed for the actual pixel buffer. These operations occur when you call the function `aglSetPBuffer` (page 40).

You can determine the dimensional limits of a pixel buffer by calling the OpenGL function `glGetInteger`. You can find the maximum size supported by querying `GL_MAX_VIEWPORT_DIMS` and the minimum size by querying `GL_MIN_PBUFFER_VIEWPORT_DIMS_APPLE`, which returns two integer values (similar to `GL_MAX_VIEWPORT_DIMS`). All pixel buffer dimensions that you request with the function `aglCreatePBuffer` should fall within these limits (inclusively) and should comply with any limitations imposed by the texture target you select.

The maximum viewport size supported in Mac OS X is quite large. You should take into consideration the amount of video or system memory required to support the requested pixel buffer size, including additional memory needed for multiple buffers and options such as multisampling.

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

See Also
`aglDescribePBuffer` (page 17)
`aglDestroyPBuffer` (page 20)

Declared In
`agl.h`

aglCreatePixelFormat

Creates a pixel format with the provided attributes.

```
AGLPixelFormat aglCreatePixelFormat (
   const GLint *attribs
);
```

Parameters

attribs
The attributes for the pixel format.

Return Value
A pixel format object.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`agl.h`

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

aglDescribePBuffer

Retrieves information that describes the specified pixel buffer object.

```
GLboolean aglDescribePBuffer (
   AGLPbuffer pbuffer,
   GLint *width,
   GLint *height,
   GLenum *target,
   GLenum *internalFormat,
   GLint *max_level
);
```

Parameters

pbuffer

A pixel buffer object.

width

On return, points to the width, in pixels, of the pixel buffer.

height

On return, points to the height, in pixels, of the pixel buffer.

target

On return, points to a constant that specifies the pixel buffer target texture type, such as:

- GL_TEXTURE_2D, a texture whose dimensions are a power of two.
- GL_TEXTURE_RECTANGLE_EXT, a texture whose dimensions are not a power of two.
- GL_TEXTURE_CUBE_MAP, a mapped cube texture.

internalFormat

On return, points to a constant that specifies the internal color format of the pixel buffer— either GL_RGB or GL_RGBA.

max_level

On return, points to the maximum level of mipmap detail or 0 if the pixel buffer doesn't use mipmaps.

Return Value

Returns GL_FALSE if it fails for any reason, GL_TRUE otherwise. If an error occurs, call the function aglGetError (page 28) to determine what the error is.

Discussion

The width, height, texture target, and internal texture color format of a pixel buffer object are set at its creation and cannot be changed without destroying and then creating the object again. The level is set when the pixel buffer object is attached to a rendering context by calling the function aglSetPBuffer (page 40).

Availability

Available in Mac OS X v10.3 and later.

See Also

aglCreatePBuffer (page 15)

Declared In

agl.h

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

aglDescribePixelFormat

Retrieves the value of an attribute associated with a pixel format object.

```
GLboolean aglDescribePixelFormat (
   AGLPixelFormat pix,
   GLint attrib,
   GLint *value
);
```

Parameters

pix

The pixel format object to query.

attrib

The attribute whose value you want to obtain. For a list of possible attributes, see "Buffer and Renderer Attributes" (page 52).

value

On return, points to the value of the attribute.

Return Value

Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Discussion

On multiscreen systems that support multiple renderers simultaneously, `aglChoosePixelFormat` (page 11) can return a list of more than one pixel format object. To retrieve the data in pixel format objects other than the first one in the list, call `aglNextPixelFormat` (page 32). Then pass the returned pixel format object to `aglDescribePixelFormat` to retrieve an attribute value.

Availability

Available in Mac OS X v10.0 and later.

See Also

`aglCreateContext` (page 14)

Related Sample Code

GLCarbon1ContextPbuffer

GLCarbonSharedPbuffer

Declared In

`agl.h`

aglDescribeRenderer

Obtains the value associated with a renderer property.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
GLboolean aglDescribeRenderer (
   AGLRendererInfo rend,
   GLint prop,
   GLint *value
);
```

Parameters

rend

An opaque renderer information object that contains a description of the renderer capabilities you want to inspect. You obtain a renderer information object by calling the function `aglQueryRendererInfo` (page 34).

prop

The renderer property whose value you want to obtain. See "Renderer Properties" (page 74) for a list of the constants you can supply for this parameter. There are also a number of constants described in "Buffer and Renderer Attributes" (page 52) that you can supply to this function.

value

On return, points to the value of the requested property.

Return Value

Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Availability

Available in Mac OS X v10.0 and later.

See Also

`aglNextRendererInfo` (page 33)

`aglDestroyRendererInfo` (page 21)

Related Sample Code

OpenGLMovieQT

Declared In

`agl.h`

aglDestroyContext

Frees the resources associated with a rendering context.

```
GLboolean aglDestroyContext (
   AGLContext ctx
);
```

Parameters

ctx

The rendering context whose resources you want to release.

Return Value

Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Discussion
This function frees all the resources used by the rendering context passed to it. If the rendering context that you pass is the current rendering context, the current context is set to `NULL` and there is no current rendering context after the function executes.

After you call this function, you must make sure that you do not use the destroyed rendering context. This includes using AGL macros in which the rendering context is explicitly passed to OpenGL.

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

See Also
`aglCreateContext` (page 14)
`aglUpdateContext` (page 46)

Related Sample Code
AGLSurfaceTexture
Carbon GLSnapshot
GLCarbon1ContextPbuffer
GLCarbonSharedPbuffer
OpenGLMovieQT

Declared In
`agl.h`

aglDestroyPBuffer

Releases the resources associated with a pixel buffer object.

```
GLboolean aglDestroyPBuffer (
   AGLPbuffer pbuffer
);
```

Parameters

pbuffer
The pixel buffer object whose resources you want to release.

Return Value
Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Discussion
Call this function only after you no longer need to use the pixel buffer object. Before calling this function, you should delete all texture objects associated with the pixel buffer object. You do not need to make sure that all texturing commands have completed prior to calling this function, because the OpenGL framework manages texturing synchronization.

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

See Also
`aglCreatePBuffer` (page 15)

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Related Sample Code
GLCarbon1ContextPbuffer
GLCarbonSharedPbuffer

Declared In
`agl.h`

aglDestroyPixelFormat

Frees the memory associated with a pixel format object.

```
void aglDestroyPixelFormat (
   AGLPixelFormat pix
);
```

Parameters

pix

The pixel format object whose resources you want to release. This must be a pixel format object returned by the function`aglChoosePixelFormat` (page 11). AGL sets a `AGL_BAD_PIXELFMT` error if you pass the returned value from the function `aglNextPixelFormat` (page 32) or the `pix` parameter is not a valid pixel format.

Discussion
A copy of the pixel format data is made by the `aglCreateContext` (page 14) function, so your application can free a pixel format immediately after creating a rendering context with it. On the other hand, a pixel format object can be useful for enumerating virtual screens when multiple renderers are supported, so you may want to retain it.

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

See Also
`aglDescribePixelFormat` (page 18)

Related Sample Code
AGLSurfaceTexture
Carbon GLSnapshot
GLCarbon1ContextPbuffer
GLCarbonSharedPbuffer
OpenGLMovieQT

Declared In
`agl.h`

aglDestroyRendererInfo

Frees resources associated with a renderer information object.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
void aglDestroyRendererInfo (
   AGLRendererInfo rend
);
```

Parameters

rend

The renderer information object whose resources you want to release.

Discussion

AGL sets a `AGL_BAD_RENDINFO` error if you pass an invalid renderer information object. You can check for this error by calling aglGetError (page 28).

Availability

Available in Mac OS X v10.0 and later.

See Also

`aglQueryRendererInfo` (page 34)

Related Sample Code

OpenGLMovieQT

Declared In

`agl.h`

aglDevicesOfPixelFormat

Returns the graphics devices supported by a pixel format object. (**Deprecated.** Use `aglDisplaysOfPixelFormat` (page 23) instead.)

```
GDHandle * aglDevicesOfPixelFormat (
   AGLPixelFormat pix,
   GLint *ndevs
);
```

Parameters

pix

A pixel format object.

ndevs

On return, points to the number of devices in the list returned by the function.

Return Value

An array of graphics devices or or `NULL` if the function fails for any reason. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Discussion

You typically use this function after you call the function `aglChoosePixelFormat` (page 11), which can return a list of more than one pixel format object. The first pixel format object in the list is guaranteed to support all of the graphics devices requested of that function.

Availability

Available in Mac OS X v10.0 and later.

See Also

`aglDescribePixelFormat` (page 18)

`aglNextPixelFormat` (page 32)

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Declared In
agl.h

aglDisable

Disables an option for a rendering context.

```
GLboolean aglDisable (
   AGLContext ctx,
   GLenum pname
);
```

Parameters

ctx

A rendering context.

pname

The capability that you want to disable. You can pass any of the constants listed in "Context Options and Parameters" (page 63) that specify they can be enabled or disabled.

Return Value
Returns GL_FALSE if it fails for any reason, GL_TRUE otherwise. If an error occurs, call the function aglGetError (page 28) to determine what the error is.

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

See Also
aglEnable (page 24)

Related Sample Code
aglClipBufferRect
OpenGLMovieQT

Declared In
agl.h

aglDisplaysOfPixelFormat

Returns the graphics devices supported by a pixel format object.

```
CGDirectDisplayID * aglDisplaysOfPixelFormat (
   AGLPixelFormat pix,
   GLint *ndevs
);
```

Parameters

pix

A pixel format object.

ndevs

On return, points to the number of devices in the list returned by the function.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Return Value
An array of display IDs or or `NULL` if the function fails for any reason. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Discussion
You typically use this function after you call the function `aglChoosePixelFormat` (page 11), which can return a list of more than one pixel format object. The first pixel format object in the list is guaranteed to support all of the graphics devices requested of that function. You call `aglDisplaysOfPixelFormat` to find out which graphics devices are supported by other pixel format objects in the list.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`agl.h`

aglEnable

Enables an option for a rendering context.

```
GLboolean aglEnable (
   AGLContext ctx,
   GLenum pname
);
```

Parameters

ctx

A rendering context.

pname

The option you want to enable. You can pass any of the constants listed in "Context Options and Parameters" (page 63) that specify they can be enabled or disabled.

Return Value
Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Discussion
To set the value associated with a rendering context option, use the function `aglSetInteger` (page 38). For example, if you enable `AGL_SWAP_RECT`, you can specify the area of the window that is affected by the `aglSwapBuffers` (page 44) function by setting the rectangle size with the function `aglSetInteger` (page 38).

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

See Also
`aglDisable` (page 23)
`aglIsEnabled` (page 32)

Related Sample Code
aglClipBufferRect
OpenGLMovieQT

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Declared In
agl.h

aglErrorString

Returns a string that describes the specified AGL error code.

```
const GLubyte * aglErrorString (
   GLenum code
);
```

Parameters

code

An AGL error code returned by the function aglGetError (page 28). For a description of these constants, see "Error Codes" (page 66).

Return Value
An error string that describes the error code passed in the *code* parameter. If the code is invalid, returns the string "No such error code."

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

See Also
aglGetError (page 28)

Related Sample Code
AGLSurfaceTexture
GLCarbon1ContextPbuffer
GLCarbonSharedPbuffer
OpenGLMovieQT

Declared In
agl.h

aglGetCGLContext

Gets the CGL rendering context associated with an AGL rendering context.

```
GLboolean aglGetCGLContext (
   AGLContext ctx,
   void **cgl_ctx
);
```

Parameters

ctx

An AGL rendering context.

cgl_ctx

On return, points to the CGL rendering context associated with the specified AGL rendering context.

Return Value
Returns GL_FALSE if it fails for any reason, GL_TRUE otherwise. If an error occurs, call the function aglGetError (page 28) to determine what the error is.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Discussion
You should use this function only when you absolutely need to use a non-AGL function that requires an CGL rendering context. You can cast the returned CGL rendering context to another data type as needed. Note that destroying the AGL rendering context also destroys the CGL rendering context, thus rendering the data reference returned by this function invalid.

Note: Always use the AGL version of a function when available, because there is additional work that AGL must perform to be compatible with Carbon.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
QTCarbonCoreImage101

Declared In
`agl.h`

aglGetCGLPixelFormat

Gets the CGL pixel format object associated with an AGL pixel format.

```
GLboolean aglGetCGLPixelFormat (
   AGLPixelFormat pix,
   void **cgl_pix
);
```

Parameters

pix
An AGL pixel format object.

cgl_pix
On return, points to the CGL pixel format object associated with the specified AGL pixel format.

Return Value
Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Discussion
You should use this function only when you absolutely need to pass a CGL pixel format object to a non-AGL function. You can cast the returned CGL pixel format object to another data type as needed. Note that destroying the AGL pixel format object also destroys the CGL pixel format object, thus making the data reference returned invalid.

Note: Always use the AGL version of a function when available, because there is additional work that AGL must perform to be compatible with Carbon.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
QTCarbonCoreImage101

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Declared In
agl.h

aglGetCurrentContext

Returns the current rendering context.

```
AGLContext aglGetCurrentContext (
   void
);
```

Return Value
The current rendering context. If there is no current rendering context, returns NULL.

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

See Also
aglSetCurrentContext (page 35)

Declared In
agl.h

aglGetDrawable

Returns the drawable object attached to a rendering context.

```
AGLDrawable aglGetDrawable (
   AGLContext ctx
);
```

Parameters
ctx
A rendering context that's attached to a window or a graphics device.

Return Value
The drawable object (either a Carbon window or a full-screen graphics port) that is attached to the rendering context, or NULL for any of the following reasons:

- No drawable object is attached.
- A pixel buffer or offscreen memory area is attached.
- The function fails for any reason.

If the function returns NULL, call the function aglGetError (page 28) to determine what the error is.

Discussion
If the rendering context is a pixel buffer context, call aglGetPBuffer (page 29).

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

See Also
aglSetDrawable (page 36)

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Declared In
agl.h

aglGetError

Returns an AGL error code.

```
GLenum aglGetError (
   void
);
```

Return Value
The value of the global AGL error flag. See "Error Codes" (page 66) for a complete description of the error codes that can be returned.

Discussion
This function is similar to the OpenGL function `glGetError`. You call the function `aglGetError` whenever an AGL function returns `FALSE` to retrieve the error code associated with the error condition. Each error is assigned a numeric code and a symbolic name. When an error occurs, the error flag is set to the appropriate error code value. No other errors are recorded until `aglGetError` is called, the error code is returned, and the flag is reset to `AGL_NO_ERROR`. If a call to `aglGetError` returns `AGL_NO_ERROR`, this means that there has been no detectable error since the last call to `aglGetError`.

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

See Also
`aglErrorString` (page 25)

Related Sample Code
AGLSurfaceTexture
GLCarbon1ContextPbuffer
GLCarbonSharedPbuffer
OpenGLMovieQT
QTCarbonCoreImage101

Declared In
agl.h

aglGetHIViewRef

Retrieves the HIView object associated with an AGL context.

```
HIViewRef aglGetHIViewRef (
   AGLContext ctx
);
```

Parameters
ctx
The AGL context whose HIView object you want to retrieve.

Return Value
The HIView object associated with the context.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`agl.h`

aglGetInteger

Retrieves the integer setting of an AGL context option.

```
GLboolean aglGetInteger (
   AGLContext ctx,
   GLenum pname,
   GLint *params
);
```

Parameters

ctx

An rendering context.

pname

The option whose value you want to retrieve. You can pass any of the constants listed in "Context Options and Parameters" (page 63) that have an associated value.

params

On return, points to the value of the option.

Return Value
Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is. See "Error Codes" (page 66) for more information.

Discussion
Use `aglEnable` (page 24) to enable the option.

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

See Also
`aglSetInteger` (page 38)

Declared In
`agl.h`

aglGetPBuffer

Retrieves a pixel buffer and its parameters for a specified rendering context.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
GLboolean aglGetPBuffer (
   AGLContext ctx,
   AGLPbuffer *pbuffer,
   GLint *face,
   GLint *level,
   GLint *screen
);
```

Parameters

ctx

A rendering context.

pbuffer

On return, points to the pixel buffer object attached to the rendering context.

face

On return, points to the cube map face that is set if the pixel buffer texture target type is `GL_TEXTURE_CUBE_MAP`; otherwise `0` for all other texture target types.

level

On return, points to the current mipmap level for drawing.

screen

On return, points to the current virtual screen number, as set by the last valid call to aglSetPBuffer (page 40).

Return Value

Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Availability

Available in Mac OS X v10.3 and later.

See Also

`aglSetPBuffer` (page 40)

Declared In

`agl.h`

aglGetVersion

Gets the major and minor version numbers of the AGL library.

```
void aglGetVersion (
   GLint *major,
   GLint *minor
);
```

Parameters

major

On return, points to he major version number of the AGL library.

minor

On return, points to the minor version number of the AGL library.

Discussion

AGL implementations with the same major version number are upwardly compatible, meaning that the implementation with the highest minor number is a superset of the version with the lowest minor number.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Certain features, such as index color support, are deprecated in Mac OS X. The current version of AGL might not be a full superset of AGL for Mac OS 9 or of earlier versions of AGL.

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

Declared In
`agl.h`

aglGetVirtualScreen

Returns the current virtual screen number associated with a rendering context.

```
GLint aglGetVirtualScreen (
   AGLContext ctx
);
```

Parameters

ctx
A rendering context.

Return Value
Returns the virtual screen number of the context. The value is always `0` on a single-monitor system and `-1` if the function fails for any reason. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Discussion
The current virtual screen is set automatically by the function `aglUpdateContext` (page 46). The current virtual screen can change when a drawable object is moved or resized across graphics device boundaries. A change in the current virtual screen can affect the return values of some OpenGL functions and in most cases also means that the renderer has changed.

For detailed information on virtual screens, see *OpenGL Programming Guide for Mac OS X*.

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

See Also
`aglSetVirtualScreen` (page 41)

Related Sample Code
GLCarbon1ContextPbuffer
GLCarbonSharedPbuffer

Declared In
`agl.h`

aglGetWindowRef

Retrieves the window associated with an AGL context.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
WindowRef aglGetWindowRef (
   AGLContext ctx
);
```

Parameters

ctx

The AGL context whose window you want to retrieve.

Return Value
The window associated with the context.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`agl.h`

aglIsEnabled

Reports whether an option is enabled for a rendering context.

```
GLboolean aglIsEnabled (
   AGLContext ctx,
   GLenum pname
);
```

Parameters

ctx

A rendering context.

pname

The capability whose state you want to check. You can pass any of the constants listed in "Context Options and Parameters" (page 63) that specify they can be enabled or disabled.

Return Value
`GL_FALSE` if the option is disabled or if it fails for any reason; `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

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

See Also
`aglEnable` (page 24)
`aglDisable` (page 23)

Declared In
`agl.h`

aglNextPixelFormat

Returns the next pixel format object in a list of pixel format objects.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
AGLPixelFormat aglNextPixelFormat (
   AGLPixelFormat pix
);
```

Parameters

pix

A pointer to pixel format object.

Return Value

The next pixel format object in a list of pixel formats. Returns `NULL` if the *pix* parameter represents the last pixel format object in the list or if the function fails for any reason. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Discussion

You typically use this function after you've called the function `aglChoosePixelFormat` (page 11), which generates a list of more than one pixel format object when all the graphics devices on the system are not supported by a single renderer. To find the number of renderers associated with a pixel format object, you can call `aglNextPixelFormat` to iterate through the list and count the entries.

Availability

Available in Mac OS X v10.0 and later.

See Also

`aglDescribePixelFormat` (page 18)

Related Sample Code

GLCarbon1ContextPbuffer

GLCarbonSharedPbuffer

Declared In

`agl.h`

aglNextRendererInfo

Returns the next renderer information object.

```
AGLRendererInfo aglNextRendererInfo (
   AGLRendererInfo rend
);
```

Parameters

rend

A renderer information object that contains one or more renderer information objects. You obtain a renderer information object by calling the function `aglQueryRendererInfo` (page 34).

Return Value

The next renderer information object in the list. Returns `NULL` if the object is the last in the list or if the function fails for any reason. In case of a failure, call the function `aglGetError` (page 28) to determine what the error is.

Discussion

You typically use this function to iterate through a list of renderer information objects returned by the function `aglQueryRendererInfo` (page 34). Most systems have more than one renderer installed because support for different buffer depths is typically provided by separate renderers.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

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

See Also
aglDescribeRenderer (page 18)
aglDestroyRendererInfo (page 21)

Related Sample Code
OpenGLMovieQT

Declared In
agl.h

aglQueryRendererInfo

Creates and returns a renderer information object that contains properties and values for all renderers driving the specified displays. (**Deprecated.** Instead use aglQueryRendererInfoForCGDirectDisplayIDs (page 35).)

```
AGLRendererInfo aglQueryRendererInfo (
   const AGLDevice *gdevs,
   GLint ndev
);
```

Parameters

gdevs

A pointer to a list of graphics devices to which you want to restrict the query. Pass NULL to obtain information for all graphics devices on the system.

ndev

The number of graphics devices in *gdev*. Pass 0 if gdevs is NULL. AGL sets the AGL_BAD_DEVICE error if the ndev parameter is nonzero and the gdevs parameter is not an array of valid devices.

Return Value
A renderer information object that describes all renderers able to drive the devices specified by the gdevs parameter. The function returns NULL if it fails for any reason. If an error occurs, call the function aglGetError (page 28) to determine what the error is.

Discussion
This function normally returns a list of more than one renderer information object—one for each renderer found on the system. To access data in the first renderer information object tin the list, call the function aglDescribeRenderer (page 18). To access data in a renderer information object that is not the first one in the list, use aglNextRendererInfo (page 33).

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

See Also
aglDestroyRendererInfo (page 21)

Related Sample Code
OpenGLMovieQT

Declared In
agl.h

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

aglQueryRendererInfoForCGDirectDisplayIDs

Creates and returns a renderer information object that contains properties and values for all renderers driving the specified displays.

```
AGLRendererInfo aglQueryRendererInfoForCGDirectDisplayIDs(
    const CGDirectDisplayID *dspIDs,
    GLint ndev
);
```

Parameters

dspIDs

A pointer to the list of IDs to which you want to restrict the query. . Pass `NULL` to obtain information for all graphics devices on the system.

ndev

The number of graphics devices in *dspIDs*. Pass `0` if `dspIDs` is `NULL`. AGL sets the `AGL_BAD_DEVICE` error if the `ndev` parameter is nonzero and the `dspIDs` parameter is not an array of valid devices.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`agl.h`

aglResetLibrary

Resets the AGL library. (**Deprecated.** This function is not needed in Mac OS X.)

```
void aglResetLibrary (
   void
);
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`agl.h`

aglSetCurrentContext

Sets the specified rendering context as the current rendering context.

```
GLboolean aglSetCurrentContext (
   AGLContext ctx
);
```

Parameters

ctx

The rendering context to set as the current rendering context. Pass `NULL` to release the current rendering context without assigning a new one.

Return Value

Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Discussion
There can be only one current rendering context. Subsequent OpenGL rendering calls operate on the current rendering context to modify the drawable object associated with it.

You can use AGL macros to bypass the current rendering context mechanism and maintain your own current rendering context.

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

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

See Also
`aglGetCurrentContext` (page 27)

Related Sample Code
AGLSurfaceTexture
Carbon GLSnapshot
GLCarbon1ContextPbuffer
GLCarbonSharedPbuffer
OpenGLMovieQT

Declared In
`agl.h`

aglSetDrawable

Attaches a rendering context to a Carbon window.

```
GLboolean aglSetDrawable (
   AGLContext ctx,
   AGLDrawable draw
);
```

Parameters

ctx

A rendering context returned by the function aglCreateContext (page 14).

draw

The drawable object—which must be a Carbon window—to attach to the AGL rendering context. You need to supply a `CGrafPtr` data type obtained from a valid window. The Window Manager function `GetWindowPort` returns the `CGrafPtr` associated with a Carbon window. To disable rendering for a rendering context, pass `NULL`.

Return Value
Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Discussion
After calling this function, AGL directs subsequent OpenGL rendering calls to the specified rendering context to modify the window content. In addition, the function `aglSetDrawable` performs all of the actions performed by `aglUpdateContext` (page 46).

When a rendering context is initially attached to the window, its viewport is set to the full size of the window. If the rendering context is subsequently attached to the same window, its viewport is unaltered.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

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

See Also
`aglGetDrawable` (page 27)

Related Sample Code
AGLSurfaceTexture
Carbon GLSnapshot
GLCarbon1ContextPbuffer
GLCarbonSharedPbuffer
OpenGLMovieQT

Declared In
`agl.h`

aglSetFullScreen

Attaches a rendering context to a full screen graphics device.

```
GLboolean aglSetFullScreen (
   AGLContext ctx,
   GLsizei width,
   GLsizei height,
   GLsizei freq,
   GLint device
);
```

Parameters

ctx
A rendering context.

width
The width, in pixels, of the graphics device. This value must be an exact match to the graphics device.

height
The height, in pixels, of the graphics device. This value must be an exact match to the graphics device.

freq
The refresh frequency of the graphics device, in Hertz. If you specify a *frequency* of `0`, AGL uses the highest refresh frequency available for the specified *width* and *height*. If you specify a nonzero frequency, AGL chooses the closest frequency available for the given geometry. If a display provides only a `0` refresh frequency, AGL matches it with *width* and *height* regardless of the value of *frequency* passed.

device
This parameter is ignored in Mac OS X, where all devices that are compatible with the pixel format of the rendering context are considered when selecting a full screen graphics device.

Return Value
Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Discussion
Passing `0` for the width, `0` for the height, and `0` for the frequency sets up a full screen context at the current height, width, and refresh rate.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

After calling this function, subsequent OpenGL rendering calls directed to the full-screen graphics device. The rendering context must be created with respect to a pixel format that supports a full-screen device, which you can request by passing `AGL_FULLSCREEN` to the function `aglChoosePixelFormat` (page 11). The `aglSetFullScreen` function also performs all of the actions performed by `aglUpdateContext` (page 46). The `aglSetFullScreen` function uses information obtained from the pixel format object use to create the rendering context to choose the color depth for full-screen display mode.

When a rendering context is initially attached to a full screen graphics device, its viewport is set to the full size of the device. If the rendering context is subsequently attached to the same device, its viewport is unaltered. To disable a rendering context, call `aglSetDrawable` (page 36) with the `draw` parameter set to `NULL`.

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

See Also
`aglGetDrawable` (page 27)

Declared In
`agl.h`

aglSetHIViewRef

Sets an AGL context to the specified HIView object.

```
GLboolean aglSetHIViewRef (
   AGLContext ctx,
   HIViewRef hiview
);
```

Parameters

ctx
An AGL context.

hiview
The HIView object to set.

Return Value
`true` if the HIView object is successfully set; `false` otherwise.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`agl.h`

aglSetInteger

Sets the value of an option for a rendering context.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
GLboolean aglSetInteger (
   AGLContext ctx,
   GLenum pname,
   const GLint *params
);
```

Parameters

ctx

A rendering context.

pname

The rendering context option whose value you want to set. You can pass any of the constants listed in "Context Options and Parameters" (page 63) that have an associated integer value.

params

A pointer to the value to set the parameter to.

Return Value

Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Discussion

Use `aglEnable` (page 24) to enable the option.

Availability

Available in Mac OS X v10.0 and later.

See Also

`aglGetInteger` (page 29)

Related Sample Code

aglClipBufferRect

GLCarbon1ContextPbuffer

OpenGLMovieQT

QTCarbonCoreImage101

Declared In

`agl.h`

aglSetOffScreen

Attaches a rendering context to an offscreen memory area.

```
GLboolean aglSetOffScreen (
   AGLContext ctx,
   GLsizei width,
   GLsizei height,
   GLsizei rowbytes,
   GLvoid *baseaddr
);
```

Parameters

ctx

A rendering context returned by the function aglCreateContext (page 14).

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

width

The width, in pixels, of the offscreen memory area.

height

The height, in pixels, of the offscreen memory area.

rowbytes

The number of bytes in each row of the offscreen memory area.

baseaddr

A pointer to the base address of the memory area.

Return Value

Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If it fails, call the function `aglGetError` (page 28) to determine what the error is.

Discussion

After calling this function, subsequent OpenGL rendering calls modify the offscreen memory. The rendering context must be created with respect to a pixel format that supports offscreen rendering, which you can request by passing `AGL_OFFSCREEN` to the `aglChoosePixelFormat` (page 11) function. The `aglSetOffScreen` function also performs all of the actions performed by `aglUpdateContext` (page 46). The pixel size of the pixel format is set so that it is equal to the buffer depth of the offscreen area.

When a rendering context is attached to an offscreen memory area, its viewport is set to the full size of the offscreen area.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`agl.h`

aglSetPBuffer

Attaches a pixel buffer object to a rendering context.

```
GLboolean aglSetPBuffer (
   AGLContext ctx,
   AGLPbuffer pbuffer,
   GLint face,
   GLint level,
   GLint screen
);
```

Parameters

ctx

A rendering context.

pbuffer

A pixel buffer object.

face

The cube map face to draw if the buffer texture target is type `GL_TEXTURE_CUBE_MAP`; otherwise pass `0`.

level

The mipmap level to draw. This must not exceed the maximum mipmap level set when the pixel buffer object was created. Pass `0` for a texture target that does not support mipmaps.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

screen

A virtual screen value. The virtual screen determines the renderer that OpenGL uses to draw to the pixel buffer object. For best performance, for a pixel buffer used as a texture source, you should supply the a virtual screen value that results in using the same renderer used by the context that's the texturing target.

Return Value
Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is. See Discussion for more details.

Discussion
The first time you call this function for a specific pixel buffer object, the system creates the necessary buffers. The buffers are created to support the pixel formats used to create the rendering context. The storage requirements for pixel buffer objects, which can be quite large, are very similar to the requirements for windows or views with OpenGL contexts attached. All drawable objects compete for the same scarce resources. This function can fail is there is not enough contiguous VRAM for each buffer. It's best to code defensively with a scheme that reduces resource consumption without causing the application to resort to failure. Unless, of course, failure is the only viable alternative.

The ability to attach a pbuffer to a context is supported only on renderers that export `GL_APPLE_pixel_buffer` in the `GL_EXTENSIONS` string. Before calling this function, you should programmatically determine if it's possible to attach a pbuffer to a context by querying `GL_EXTENSIONS` in the context and looking for `GL_APPLE_pixel_buffer`. If that extension is not present, the renderer won't allow setting the pbuffer. (In this case, the function correctly returns `GL_FALSE`, although the error returned by `aglGetError` (page 28) may be misleading.)

In order of performance, these are the renderers you should consider using when setting up a rendering context to attach to a pbuffer:

- A hardware renderer.
- The generic render, but only with an offscreen pixel format and `glTexSubImage`.
- The Apple software renderer, which supports pbuffers in Mac OS X v10.4.8 and later.

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

See Also
`aglGetPBuffer` (page 29)

Declared In
`agl.h`

aglSetVirtualScreen

Forces subsequent OpenGL commands to the specified virtual screen.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
GLboolean aglSetVirtualScreen (
   AGLContext ctx,
   GLint screen
);
```

Parameters

ctx

A rendering context.

screen

A virtual screen number, which must be a value between 0 and the number of virtual screens minus one. You can obtain the number of virtual screens available associated with a rendering context by calling the function aglDescribePixelFormat (page 18), passing in the pixel format object used to create the rendering context and the attribute constant AGL_VIRTUAL_SCREEN.

Return Value

Returns GL_FALSE if it fails for any reason, GL_TRUE otherwise. If an error occurs, call the function aglGetError (page 28) to determine what the error is.

Discussion

You should use the aglSetVirtualScreen function only when it is necessary to override the default behavior. The current virtual screen is normally set automatically by the functions aglSetDrawable (page 36) or aglUpdateContext (page 46). Setting the virtual screen forces the renderer associated with the virtual screen to process OpenGL commands issued to the specified context. Changing the virtual screen changes the current renderer. Because the current virtual screen determines which OpenGL renderer is processing commands, the return values of all glGetXXX functions can be affected by the current virtual screen.

For detailed information on virtual screens, see *OpenGL Programming Guide for Mac OS X*.

Availability

Available in Mac OS X v10.0 and later.

See Also

aglGetVirtualScreen (page 31)

Declared In

agl.h

aglSetWindowRef

Sets an AGL context to the specified window.

```
GLboolean aglSetWindowRef (
   AGLContext ctx,
   WindowRef window
);
```

Parameters

ctx

An AGL context.

window

The window to set.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Return Value
`true` if the window is successfully set; `false` otherwise.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`agl.h`

aglSurfaceTexture

Allows texturing from a drawable object that has an attached rendering context, using the surface contents as the source data for the texture.

```
void aglSurfaceTexture (
   AGLContext context,
   GLenum target,
   GLenum internalformat,
   AGLContext surfacecontext
);
```

Parameters

context
A rendering context attached to the window that's the target of the texture operation.

target
A constant that specifies an OpenGL target texture. You can supply any of the texture targets defined by the OpenGL specification, including:

- `GL_TEXTURE_2D`, a texture whose dimensions are a power of two.
- `GL_TEXTURE_RECTANGLE_EXT`, a texture whose dimensions are not a power of two.
- `GL_TEXTURE_CUBE_MAP`, a mapped cube texture.

internalformat
A constant that specifies the internal color format of the texture— either `GL_RGB` or `GL_RGBA`.

surfacecontext
A rendering context that's attached to the window that's the source of the texture data.

Discussion
The `aglSurfaceTexture` function behaves similar to the function `glTexImage2D`. The texture target and internal format must be supported by the renderer of the target rendering context, and the geometry of the source drawable object must be compatible with the texture target. For example, if the texture target is `GL_TEXTURE_2D`, the window must conform to power-of-two dimensions.

Before calling this function you must use OpenGL calls to set up the OpenGL texture referred to by the target parameter. That is, enable texturing, generate a texture name, bind it to a texture target, and so forth. For more information, see OpenGL Programming Guide.

The target and source windows must use the same renderer (virtual screen) or the function fails, most typically by not texturing the target rendering context.

An error condition occurs if the surface rendering context is not attached to a windowed drawable object; the drawable object cannot be an offscreen area or pixel buffer. It's also possible to get standard OpenGL errors similar to the errors that can occur for the function `glTexImage2D`. Use the function `glError` to query OpenGL errors.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Note: Although the function `aglSurfaceTexture` provides a flexible way to render to an offscreen texture and then use it as a source, you should consider using pixel buffer objects or OpenGL framebuffer objects instead. For details on using a hidden window as a texture source, see *OpenGL Programming Guide for Mac OS X*. This document also describes how to use pixel buffer objects and framebuffer objects as texture sources.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
AGLSurfaceTexture

Declared In
`agl.h`

aglSwapBuffers

Exchanges the front and back buffers of the specified rendering context.

```
void aglSwapBuffers (
   AGLContext ctx
);
```

Parameters

ctx

The rendering context whose buffers you want to swap. AGL sets the `AGL_BAD_CONTEXT` error if the `ctx` parameter is not a valid AGL rendering context.

Discussion
The `aglSwapBuffers` function either swaps the buffers or copies the back buffer content to the front buffer. Before it returns, `aglSwapBuffers` invokes the OpenGL function `glFlush`.

To synchronize with a monitor retrace, set the current swap interval (`AGL_SWAP_INTERVAL`) to `1` by calling the function `aglSetInteger` (page 38).

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

Related Sample Code
aglClipBufferRect
AGLSurfaceTexture
Carbon GLSnapshot
GLCarbon1ContextPbuffer
OpenGLMovieQT

Declared In
`agl.h`

aglTexImagePBuffer

Binds the contents of a pixel buffer to a data source for a texture object.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
GLboolean aglTexImagePBuffer (
   AGLContext ctx,
   AGLPbuffer pbuffer,
   GLint source
);
```

Parameters

ctx

A rendering context, which is the target context for the texture operation. This is the context that you plan to render content to. This is not the context attached to the pixel buffer.

pbuffer

A pixel buffer object.

source

The source buffer to texture from, which should be a valid OpenGL buffer such as `GL_FRONT` or `GL_BACK` and should be compatible with the buffer and renderer attributes that you used to create the rendering context attached to the pixel buffer. This means that the pixel buffer must possess the buffer in question for the texturing operation to succeed.

Return Value

Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Discussion

You must generate and bind a texture name (using standard OpenGL texturing calls) that is compatible with the pixel buffer texture target. Don't supply a texture object that was used previously for nonpixel buffer texturing operations unless a you first call the OpenGL function `glDeleteTextures` and then regenerate the texture name.

If you modify the content of a pixel buffer that uses mipmap levels, you must call this function again before drawing with the pixel buffer, to ensure that the content is synchronized with OpenGL. For pixel buffers without mipmaps, simply rebind to the texture object to synchronize content.

No OpenGL texturing calls that modify a pixel buffer texture content are permitted (such as `glTexSubImage2D` or `glCopyTexImage2D`) with the pixel buffer texture as the destination. It *is* permitted to use texturing commands to read data *from* a pixel buffer texture, such as `glCopyTexImage2D`, with the pixel buffer texture as the *source*. It is also legal to use OpenGL functions such as `glReadPixels` to read the contents of a pixel buffer directly through the pixel buffer context.

Note that texturing with the `aglTexImagePBuffer` function can fail to produce the intended results without error in the same way other OpenGL texturing commands can normally fail. The function fails if you do not enable the proper texture target, set an incompatible filter mode, or other conditions described in the OpenGL specification.

You don't need to share a context when a pixel buffer object is a texture source. You can use independent pixel formats and OpenGL contexts for both the pixel buffer and the target drawable object without sharing resources, and still texture using a pixel buffer in the target context.

For details on how to use a pixel buffer object as a texture source, see *OpenGL Programming Guide for Mac OS X*.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

GLCarbon1ContextPbuffer

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

GLCarbonSharedPbuffer

Declared In
agl.h

aglUpdateContext

Notifies the rendering context that the window geometry has changed.

```
GLboolean aglUpdateContext (
   AGLContext ctx
);
```

Parameters

ctx

The rendering context that needs updating.

Return Value
Returns GL_FALSE if it fails for any reason, GL_TRUE otherwise. If an error occurs, call the function aglGetError (page 28) to determine what the error is.

Discussion
You must call the aglUpdateContext function any time that the geometry of the drawable object changes. You should call this function after the completion of any move or resize action.

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

See Also
aglSetDrawable (page 36)

Related Sample Code
AGLSurfaceTexture
OpenGLMovieQT

Declared In
agl.h

aglUseFont

Creates bitmap display lists from a font.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
GLboolean aglUseFont (
   AGLContext ctx,
   GLint fontID,
   Style face,
   GLint size,
   GLint first,
   GLint count,
   GLint base
);
```

Parameters

ctx

A rendering context.

fontID

The font ID associated with the font whose character glyphs you want to use. The `fontID` value can identify any font family that is valid for passing into the `TextFont` function.

face

The font style. The `face` value can be any valid style that can be passed into the `TextFace` function.

size

The font size.

first

The index of the first glyph to use.

count

The number of glyphs to use.

base

The index of the first display list to generate.

Return Value

Returns `GL_FALSE` if it fails for any reason, `GL_TRUE` otherwise. If an error occurs, call the function `aglGetError` (page 28) to determine what the error is.

Discussion

A display list contains one or more OpenGL commands that are stored and available to be executed later by your application. The `aglUseFont` function generates a set of display lists, each of which contains a single call to the OpenGL `glBitmap` function. AGL generates names for each display list by using the value specified in the `base` parameter and increasing sequentially to `base` + `count - 1`.

The prototype for the `glBitmap` function is :

```
void glBitmap(GLsizei width,
              GLsizei height,
              GLfloat xorig,
              GLfloat yorig,
              GLfloat xmove,
              GLfloat ymove,
              const GLubyte *bitmap)
```

AGL generates parameters for each `glBitmap` command in each display list using the following rules:

- The formatting parameters—`xorig`, `yorig`, `width`, and `height`—are computed from the font metrics provided in the font as `0`, font descent – 1, font width, and font ascent + descent, respectively.
- The `xmove` parameter based on the `width` metric of the glyph.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

- The `ymove` is set to `0`.
- The glyph image is converted to the appropriate format for the `bitmap` parameter.

AGL creates empty display lists for all glyphs that you request and that are not defined in the font. For current fonts, see the file `fonts.h`.

For more information on the `glBitmap` function see the OpenGL Reference Manual. You might also want to look at the `glNewList` function, which is the function that AGL uses to generate the display lists.

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

Declared In
`agl.h`

Data Types

AGLContext

Represents a pointer to an opaque AGL context object.

```
typedef struct __AGLContextRec  *AGLContext;
```

Discussion
This data type points to a structure that AGL uses to maintain state and other information associated with an OpenGL rendering context. Use the functions described in "Managing Contexts" (page 8) and "Getting and Setting Context Options" (page 8) to create, manage, access, and free an AGL context object.

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

Declared In
`agl.h`

AGLDevice

Defines a reference to a list of graphics devices.

```
typedef GDHandle AGLDevice;
```

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

Declared In
`agl.h`

AGLDrawable

Defines an opaque data type that represents a Carbon window.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
typedef CGrafPtr AGLDrawable;
```

Discussion
This data type points to a structure that AGL uses to keep track of the information needed to transmit rendering operations from bits in memory to the onscreen pixels associated with a Carbon window. This data type is a CGrafPtr "under the hood." The Window Manager function `GetWindowPort` returns the `CGrafPtr` associated with a Carbon window.

Note: The generic term *drawable object* is not the same as the `AGLDrawable` data type, which has a very specific meaning. A drawable object can be any rendering destination (pixel buffer, window, offscreen, and so on) while the the `AGLDrawable` data type is used specifically for a window rendering destination.

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

Declared In
`agl.h`

AGLPbuffer

Represents a pointer to an opaque pixel buffer object.

```
typedef struct __AGLPBufferRec  *AGLPbuffer;
```

Discussion
This data type points to a structure that AGL uses for hardware accelerated offscreen drawing. Use the functions described in "Managing Pixel Buffers" (page 9) to create, manage, access, and free a pixel buffer object.

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

Declared In
`agl.h`

AGLPixelFormat

Represents a pointer to an opaque pixel format object.

```
typedef struct __AGLPixelFormatRec *AGLPixelFormat;
```

Discussion
This data type points to a structure that AGL uses to maintain pixel format and virtual screen information for a given set of renderer and buffer options. Use the functions described in "Managing Pixel Format Objects" (page 7) to create, manage, access, and free a pixel format object.

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

Declared In
`agl.h`

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

AGLRendererInfo

Represents a pointer to an opaque renderer information object.

```
typedef struct __AGLRendererInfoRec *AGLRendererInfo;
```

Discussion
This data type points to a structure that AGL uses to maintain information about the renderers associated with a display. Use the functions described in "Getting Renderer Information" (page 10) to create, access, and free a renderer information object.

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

Declared In
`agl.h`

Constants

Bit Depths

Define resolutions for the depth and stencil buffers.

```
#define AGL_0_BIT 0x00000001
#define AGL_1_BIT 0x00000002
#define AGL_2_BIT 0x00000004
#define AGL_3_BIT 0x00000008
#define AGL_4_BIT 0x00000010
#define AGL_5_BIT 0x00000020
#define AGL_6_BIT 0x00000040
#define AGL_8_BIT 0x00000080
#define AGL_10_BIT 0x00000100
#define AGL_12_BIT 0x00000200
#define AGL_16_BIT 0x00000400
#define AGL_24_BIT 0x00000800
#define AGL_32_BIT 0x00001000
#define AGL_48_BIT 0x00002000
#define AGL_64_BIT 0x00004000
#define AGL_96_BIT 0x00008000
#define AGL_128_BIT 0x00010000
```

Constants

`AGL_0_BIT`

A bit depth of 0.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_1_BIT`

A bit depth of 1 bit.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_2_BIT`
A bit depth of 2 bits.
Available in Mac OS X v10.0 and later.
Declared in `agl.h`.

`AGL_3_BIT`
A bit depth of 3 bits.
Available in Mac OS X v10.0 and later.
Declared in `agl.h`.

`AGL_4_BIT`
A bit depth of 4 bits.
Available in Mac OS X v10.0 and later.
Declared in `agl.h`.

`AGL_5_BIT`
A bit depth of 5 bits.
Available in Mac OS X v10.0 and later.
Declared in `agl.h`.

`AGL_6_BIT`
A bit depth of 6 bits.
Available in Mac OS X v10.0 and later.
Declared in `agl.h`.

`AGL_8_BIT`
A bit depth of 7 bits.
Available in Mac OS X v10.0 and later.
Declared in `agl.h`.

`AGL_10_BIT`
A bit depth of 10 bits.
Available in Mac OS X v10.0 and later.
Declared in `agl.h`.

`AGL_12_BIT`
A bit depth of 12 bits.
Available in Mac OS X v10.0 and later.
Declared in `agl.h`.

`AGL_16_BIT`
A bit depth of 16 bits.
Available in Mac OS X v10.0 and later.
Declared in `agl.h`.

`AGL_24_BIT`
A bit depth of 24 bits.
Available in Mac OS X v10.0 and later.
Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_32_BIT`
A bit depth of 32 bits.
Available in Mac OS X v10.0 and later.
Declared in `agl.h`.

`AGL_48_BIT`
A bit depth of 48 bits.
Available in Mac OS X v10.0 and later.
Declared in `agl.h`.

`AGL_64_BIT`
A bit depth of 64 bits.
Available in Mac OS X v10.0 and later.
Declared in `agl.h`.

`AGL_96_BIT`
A bit depth of 96 bits.
Available in Mac OS X v10.0 and later.
Declared in `agl.h`.

`AGL_128_BIT`
A bit depth of 128 bits.
Available in Mac OS X v10.0 and later.
Declared in `agl.h`.

Declared In
`agl.h`

Buffer and Renderer Attributes

Specify attributes used to create a pixel format object.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
#define AGL_NONE 0
#define AGL_ALL_RENDERERS 1
#define AGL_BUFFER_SIZE 2
#define AGL_LEVEL 3
#define AGL_RGBA 4
#define AGL_DOUBLEBUFFER 5
#define AGL_STEREO 6
#define AGL_AUX_BUFFERS 7
#define AGL_RED_SIZE 8
#define AGL_GREEN_SIZE 9
#define AGL_BLUE_SIZE 10
#define AGL_ALPHA_SIZE 11
#define AGL_DEPTH_SIZE 12
#define AGL_STENCIL_SIZE 13
#define AGL_ACCUM_RED_SIZE 14
#define AGL_ACCUM_GREEN_SIZE 15
#define AGL_ACCUM_BLUE_SIZE 16
#define AGL_ACCUM_ALPHA_SIZE 17
#define AGL_PIXEL_SIZE 50
#define AGL_MINIMUM_POLICY 51
#define AGL_MAXIMUM_POLICY 52
#define AGL_OFFSCREEN 53
#define AGL_FULLSCREEN 54
#define AGL_SAMPLE_BUFFERS_ARB 55
#define AGL_SAMPLES_ARB 56
#define AGL_AUX_DEPTH_STENCIL 57
#define AGL_COLOR_FLOAT 58
#define AGL_MULTISAMPLE 59
#define AGL_SUPERSAMPLE 60
#define AGL_SAMPLE_ALPHA 61
```

Constants

`AGL_NONE`

Used to terminate a pixel format attribute list.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_ALL_RENDERERS`

This constant is a Boolean attribute. If it is present in the attributes array, pixel format selection is open to all available renderers, including debug and special-purpose renderers that are not OpenGL compliant. Do not supply a value with this constant because its presence in the array implies `true`.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_BUFFER_SIZE`

The associated value is a nonnegative integer that specifies the number of bits per color buffer. For RGBA pixel formats, the buffer size is the sum of the red, green, blue, and alpha sizes. For color index pixel formats, the buffer size is the size of the color indexes. The smallest color index buffer of at least the specified size is preferred. Ignored if the `AGL_RGBA` attribute is asserted.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_LEVEL`

The associated value is an integer that specifies the frame buffer level of the pixel format. Positive levels correspond to frame buffers that overlay the default buffer, and negative levels correspond to frame buffers that underlay the default level. Buffer level zero corresponds to the default frame buffer of the display. Buffer level one is the first overlay frame buffer, level two the second overlay frame buffer, and so on. Negative buffer levels correspond to underlay frame buffers.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RGBA`

This constant is a Boolean attribute. If it is present in the attributes array, specifies that the color buffers store red, green, blue, and alpha values. In this case, only RGBA pixel formats are considered. Do not supply a value with this constant because its presence in the array implies `true`.

If this attribute is not present in the attributes array, color buffers store color indexes. In this case, only color index pixel formats are considered.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_DOUBLEBUFFER`

This constant is a Boolean attribute. If it is present in the attributes array, only double-buffered pixel formats are considered. Otherwise, only single-buffered pixel formats are considered. Do not supply a value with this constant because its presence in the array implies `true`.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_STEREO`

This constant is a Boolean attribute. If it is present in the attributes array, only stereo pixel formats are considered. Otherwise, only monoscopic pixel formats are considered. Do not supply a value with this constant because its presence in the array implies `true`.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_AUX_BUFFERS`

The associated value is a nonnegative integer that specifies the number of auxiliary buffers that are available. A value of `0` indicates that no auxiliary buffers exist. Pixel formats with the smallest number of auxiliary buffers that meet or exceed the specified number are preferred.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RED_SIZE`

The associated value is a nonnegative integer that specifies the number of red component bits. The value is `0` if the `AGL_RGBA` attribute is `GL_FALSE`. A red buffer that most closely matches the specified size is preferred.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_GREEN_SIZE`

The associated value is a nonnegative integer that specifies the number of green component bits. The value is `0` if the `AGL_RGBA` attribute is `GL_FALSE`. A green buffer that most closely matches the specified size is preferred.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_BLUE_SIZE`

The associated value is a nonnegative integer that specifies the number of blue component bits. The value is `0` if the `AGL_RGBA` attribute is `GL_FALSE`. A blue buffer that most closely matches the specified size is preferred.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_ALPHA_SIZE`

The associated value is a nonnegative integer that specifies the number of alpha component bits. The value is `0` if the `AGL_RGBA` attribute is `GL_FALSE`. An alpha buffer that most closely matches the specified size is preferred.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_DEPTH_SIZE`

The associated value is a nonnegative integer that specifies the number of bits in the depth buffer. A depth buffer that most closely matches the specified size is preferred.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_STENCIL_SIZE`

The associated value is a nonnegative integer that specifies the number of bits in the stencil buffer. The smallest stencil buffer of at least the specified size is preferred.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_ACCUM_RED_SIZE`

The associated value is a nonnegative integer that specifies the number of bits of red stored in the accumulation buffer. A red accumulation buffer that most closely matches the specified size is preferred.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_ACCUM_GREEN_SIZE`

The associated value is a nonnegative integer that specifies the number of bits of green stored in the accumulation buffer. A green accumulation buffer that most closely matches the specified size is preferred.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_ACCUM_BLUE_SIZE`

The associated value is a nonnegative integer that specifies the number of bits of blue stored in the accumulation buffer. A blue accumulation buffer that most closely matches the specified size is preferred.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_ACCUM_ALPHA_SIZE`

The associated value is a nonnegative integer that specifies the number of bits of alpha stored in the accumulation buffer. An alpha accumulation buffer that most closely matches the specified size is preferred.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_PIXEL_SIZE`

The associated value is a nonnegative integer that specifies the number frame buffer bits per pixel. The pixel size is the number of bits required to store each pixel in the color buffer, including unused bits. If the pixel format has an alpha channel that is stored in a separate buffer, its size is not included in the pixel size.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_MINIMUM_POLICY`

This constant is a Boolean attribute. If it is present in the attributes array, the pixel format choosing policy is altered for the color, depth, and accumulation buffers such that only buffers of size greater than or equal to the desired size are considered. Do not supply a value with this constant because its presence in the array implies `true`.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_MAXIMUM_POLICY`

This constant is a Boolean attribute. If it is present in the attributes array, the pixel format choosing policy is altered for the color, depth, and accumulation buffers such that, if a nonzero buffer size is requested, the largest available buffer is preferred. Do not supply a value with this constant because its presence in the array implies `true`.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_OFFSCREEN`

This constant is a Boolean attribute. If it is present in the attributes array, only renderers that are capable of rendering to an off-screen memory area and have buffer depth exactly equal to the desired buffer depth are considered. Do not supply a value with this constant because its presence in the array implies `true`, and thereby enables the attribute.

If enabled, and if the drawable object currently attached to a rendering context is an offscreen drawable object, the associated values are the width, height, and row bytes of the offscreen memory area. If enabled, and if the drawable object is not an offscreen type, the width, height, and row bytes are each `0`. When the `AGL_OFFSCREEN` attribute is present, the `AGL_CLOSEST_POLICY` attribute is implied.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_FULLSCREEN`

This constant is a Boolean attribute. If it is present in the attributes array, only renderers that are capable of rendering to a full-screen drawable object are considered. Furthermore, the `gdevs` parameter must be set to a pointer to the `GDevice` on which full-screen rendering is desired and the `ndev` parameter must be set to `1`.

If present, and if the drawable object that is currently attached to a rendering context is a full-screen drawable object, the associated values are the width, height, and refresh frequency of the full-screen device, rounded to the nearest integer. If present, and if the drawable object is not a full-screen type, the width, height, and refresh frequency are each `0`.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_SAMPLE_BUFFERS_ARB`

The associated value is the number of multisample buffers.

Available in Mac OS X v10.2 and later.

Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_SAMPLES_ARB`
: The associated value is the number of samples per multisample buffer.

  Available in Mac OS X v10.2 and later.

  Declared in `agl.h`.

`AGL_AUX_DEPTH_STENCIL`
: The associated value is the independent depth and/or the stencil buffers for the auxiliary buffer.

  Available in Mac OS X v10.2 and later.

  Declared in `agl.h`.

`AGL_COLOR_FLOAT`
: This constant is a Boolean attribute. If it is present in the attributes array, color buffers store floating-point pixels. Do not supply a value with this constant because its presence in the array implies `true`.

  Available in Mac OS X v10.3 and later.

  Declared in `agl.h`.

`AGL_MULTISAMPLE`
: This constant is a Boolean attribute. If it is present in the attributes array, specifies a hint to the driver to prefer multisampling. Do not supply a value with this constant because its presence in the array implies `true`.

  Available in Mac OS X v10.3 and later.

  Declared in `agl.h`.

`AGL_SUPERSAMPLE`
: This constant is a Boolean attribute. If it is present in the attributes array, specifies a hint to the driver to prefer supersampling. Do not supply a value with this constant because its presence in the array implies `true`.

  Available in Mac OS X v10.3 and later.

  Declared in `agl.h`.

`AGL_SAMPLE_ALPHA`
: This constant is a Boolean attribute. If it is present in the attributes array, request alpha filtering when multisampling. Do not supply a value with this constant because its presence in the array implies `true`.

  Available in Mac OS X v10.3 and later.

  Declared in `agl.h`.

Discussion
Each of these constants can be assigned to the the attribute array passed to the function aglChoosePixelFormat (page 11). They can also be passed to the function aglDescribePixelFormat (page 18).

Declared In
`agl.h`

Buffer Mode Flags

Define constants used to set buffer modes.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
#define AGL_MONOSCOPIC_BIT 0x00000001
#define AGL_STEREOSCOPIC_BIT 0x00000002
#define AGL_SINGLEBUFFER_BIT 0x00000004
#define AGL_DOUBLEBUFFER_BIT 0x00000008
```

Constants

`AGL_MONOSCOPIC_BIT`

A left buffer.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_STEREOSCOPIC_BIT`

A left and right buffer.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_SINGLEBUFFER_BIT`

A front buffer.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_DOUBLEBUFFER_BIT`

A front and back buffer.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

Declared In

`agl.h`

Color Modes

Specify formats and color channel layout information for the color buffer.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
#define AGL_RGB8_BIT 0x00000001
#define AGL_RGB8_A8_BIT 0x00000002
#define AGL_BGR233_BIT 0x00000004
#define AGL_BGR233_A8_BIT 0x00000008
#define AGL_RGB332_BIT 0x00000010
#define AGL_RGB332_A8_BIT 0x00000020
#define AGL_RGB444_BIT 0x00000040
#define AGL_ARGB4444_BIT 0x00000080
#define AGL_RGB444_A8_BIT 0x00000100
#define AGL_RGB555_BIT 0x00000200
#define AGL_ARGB1555_BIT 0x00000400
#define AGL_RGB555_A8_BIT 0x00000800
#define AGL_RGB565_BIT 0x00001000
#define AGL_RGB565_A8_BIT 0x00002000
#define AGL_RGB888_BIT 0x00004000
#define AGL_ARGB8888_BIT 0x00008000
#define AGL_RGB888_A8_BIT 0x00010000
#define AGL_RGB101010_BIT 0x00020000
#define AGL_ARGB2101010_BIT 0x00040000
#define AGL_RGB101010_A8_BIT 0x00080000
#define AGL_RGB121212_BIT 0x00100000
#define AGL_ARGB12121212_BIT 0x00200000
#define AGL_RGB161616_BIT 0x00400000
#define AGL_ARGB16161616_BIT 0x00800000
#define AGL_INDEX8_BIT 0x20000000
#define AGL_INDEX16_BIT 0x40000000
#define AGL_RGBFLOAT64_BIT 0x01000000
#define AGL_RGBAFLOAT64_BIT 0x02000000
#define AGL_RGBFLOAT128_BIT 0x04000000
#define AGL_RGBAFLOAT128_BIT 0x08000000
#define AGL_RGBFLOAT256_BIT 0x01000000
#define AGL_RGBAFLOAT256_BIT 0x02000000
```

Constants

`AGL_RGB8_BIT`

Specifies a format that has 8 bits per pixel with an RGB channel layout: RGB=7:0, inverse color map.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RGB8_A8_BIT`

Specifies an 8-8 ARGB bits per pixel format and the channels located in the following bits: A=7:0, RGB=7:0, inverse color map.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_BGR233_BIT`

Specifies a format that has 8 bits per pixel with an RGB channel layout, and the channels located in the following bits: B=7:6, G=5:3, R=2:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_BGR233_A8_BIT`

Specifies an 8-8 ARGB bits per pixel format and the channels located in the following bits: A=7:0, B=7:6, G=5:3, R=2:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RGB332_BIT`

Specifies a format that has 8 bits per pixel with an RGB channel layout, and the channels located in the following bits: R=7:5, G=4:2, B=1:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RGB332_A8_BIT`

Specifies an 8-8 ARGB bits per pixel format and the channels located in the following bits: A=7:0, R=7:5, G=4:2, B=1:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RGB444_BIT`

Specifies a format that has 16 bits per pixel with an RGB channel layout, and the channels located in the following bits: R=11:8, G=7:4, B=3:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_ARGB4444_BIT`

Specifies a format that has 16 bits per pixel with an ARGB channel layout, and the channels located in the following bits: A=15:12, R=11:8, G=7:4, B=3:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RGB444_A8_BIT`

Specifies a format that has 8-16 bits per pixel with an ARGB channel layout, and the channels located in the following bits: A=7:0, R=11:8, G=7:4, B=3:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RGB555_BIT`

Specifies a format that has 16 bits per pixel with an RGB channel layout, and the channels located in the following bits: R=14:10, G=9:5, B=4:0. The top bit is not used.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_ARGB1555_BIT`

Specifies a format that has 16 bits per pixel with an ARGB channel layout, and the channels located in the following bits: A=15, R=14:10, G=9:5, B=4:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_RGB555_A8_BIT`

Specifies a format that has 8-16 bits per pixel with an ARGB channel layout, and the channels located in the following bits: A=7:0, R=14:10, G=9:5, B=4:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RGB565_BIT`

Specifies a format that has 16 bits per pixel with an RGB channel layout, and the channels located in the following bits: R=15:11, G=10:5, B=4:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RGB565_A8_BIT`

Specifies a format that has 8-16 bits per pixel with an ARGB channel layout, and the channels located in the following bits: A=7:0, R=15:11, G=10:5, B=4:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RGB888_BIT`

Specifies a format that has 32 bits per pixel with an RGB channel layout, and the channels located in the following bits: R=23:16, G=15:8, B=7:0; R, G, and B take 1 byte each.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_ARGB8888_BIT`

Specifies a format that has 32 bits per pixel with an ARGB channel layout, and the channels located in the following bits: A=31:24, R=23:16, G=15:8, B=7:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RGB888_A8_BIT`

Specifies a format that has 8-32 bits per pixel with an ARGB channel layout, and the channels located in the following bits: A=7:0, R=23:16, G=15:8, B=7:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RGB101010_BIT`

Specifies a format that has 32 bits per pixel with an RGB channel layout, and the channels located in the following bits: R=29:20, G=19:10, B=9:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_ARGB2101010_BIT`

Specifies a format that has 32 bits per pixel with an ARGB channel layout, and the channels located in the following bits: A=31:30 R=29:20, G=19:10, B=9:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_RGB101010_A8_BIT`

Specifies a format that has 8-32 bits per pixel with an ARGB channel layout, and the channels located in the following bits: A=7:0 R=29:20, G=19:10, B=9:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RGB121212_BIT`

Specifies a format that has 48 bits per pixel with an RGB channel layout, and the channels located in the following bits: R=35:24, G=23:12, B=11:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_ARGB12121212_BIT`

Specifies a format that has 48 bits per pixel with an ARGB channel layout, and the channels located in the following bits: A=47:36, R=35:24, G=23:12, B=11:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RGB161616_BIT`

Specifies a format that has 64 bits per pixel with an RGB channel layout, and the channels located in the following bits: R=47:32, G=31:16, B=15:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_ARGB16161616_BIT`

Specifies a format that has 64 bits per pixel with an ARGB channel layout, and the channels located in the following bits: A=63:48, R=47:32, G=31:16, B=15:0

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_INDEX8_BIT`

Specifies an 8 bit color look up table. (Deprecated)

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_INDEX16_BIT`

Specifies an 16 bit color look up table. (Deprecated)

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RGBFLOAT64_BIT`

Specifies a format that has 64 bits per pixel with an RGB channel layout, half-floating point values. (A half-float is a 16-bit floating-point value.)

Available in Mac OS X v10.3 and later.

Declared in `agl.h`.

`AGL_RGBAFLOAT64_BIT`

Specifies a format that has 64 bits per pixel with an ARGB channel layout, half-floating point values. (A half-float is a 16-bit floating-point value.)

Available in Mac OS X v10.3 and later.

Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_RGBFLOAT128_BIT`

Specifies a format that has 128 bits per pixel with an RGB channel layout, IEEE floating point values.

Available in Mac OS X v10.3 and later.

Declared in `agl.h`.

`AGL_RGBAFLOAT128_BIT`

Specifies a format that has 128 bits per pixel with an ARGB channel layout, IEEE floating point values.

Available in Mac OS X v10.3 and later.

Declared in `agl.h`.

`AGL_RGBFLOAT256_BIT`

Specifies a format that has 256 bits per pixel with an RGB channel layout, IEEE double values.

Available in Mac OS X v10.3 and later.

Declared in `agl.h`.

`AGL_RGBAFLOAT256_BIT`

Specifies a format that has 256 bits per pixel with an ARGB channel layout, IEEE double values.

Available in Mac OS X v10.3 and later.

Declared in `agl.h`.

Declared In

`agl.h`

Context Options and Parameters

Define options and parameters that apply to a specific rendering context.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
#define AGL_SWAP_RECT 200
#define AGL_BUFFER_RECT 202
#define AGL_SWAP_LIMIT 203
#define AGL_COLORMAP_TRACKING 210
#define AGL_COLORMAP_ENTRY 212
#define AGL_RASTERIZATION 220
#define AGL_SWAP_INTERVAL 222
#define AGL_STATE_VALIDATION 230
#define AGL_BUFFER_NAME 231
#define AGL_ORDER_CONTEXT_TO_FRONT 232
#define AGL_CONTEXT_SURFACE_ID 233
#define AGL_CONTEXT_DISPLAY_ID 234
#define AGL_SURFACE_ORDER 235
#define AGL_SURFACE_OPACITY 236
#define AGL_CLIP_REGION 254
#define AGL_FS_CAPTURE_SINGLE 255
#define AGL_SURFACE_BACKING_SIZE 304
#define AGL_ENABLE_SURFACE_BACKING_SIZE 305
#define AGL_SURFACE_VOLATILE 306
```

Constants

`AGL_SWAP_RECT`

Enable or set the swap rectangle. The associated parameter must contain four values: the *x* and *y* window coordinates of the swap rectangle, followed by its width and height. If enabled, the area of the window that is affected by the function `aglSwapBuffers` (page 44) is restricted to a subrectangle of the entire window.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_BUFFER_RECT`

Enable or set the buffer rectangle. The associated parameter must contain four values: the *x* and *y* window coordinates of the buffer rectangle, relative to the structure bounds of the window, followed by its width and height. If enabled, the drawable rectangle of the window and all of its associated buffers are restricted to the specified rectangle.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_SWAP_LIMIT`

Enable or disable the swap asynchronous limit.

Available in Mac OS X v10.2 and later.

Declared in `agl.h`.

`AGL_COLORMAP_TRACKING`

Enable or disable color map tracking.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_COLORMAP_ENTRY`

The associated value is a color map entry specifies as {`index, r, g, b`}.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_RASTERIZATION`

Enable or disable all rasterization of 2D and 3D primitives. You can use this option to debug and characterize the performance of an OpenGL driver without actually rendering.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_SWAP_INTERVAL`

The associated parameter contains one value: the current swap interval setting. A value of `0` specifies not to synchronize to the vertical retrace. All other values indicate to synchronize to the vertical retrace.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_STATE_VALIDATION`

Enables or disables state validation for multiscreen functionality. If enabled, the AGL library inspects the rendering context state each time that the function `aglUpdateContext` (page 46) 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 option is useful when you want to use a single monitor system to test that an application will perform correctly on a multiple monitor system.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_BUFFER_NAME`

The associated value is a buffer name. You can use this option to allow multiple OpenGL contexts to share a buffer.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_ORDER_CONTEXT_TO_FRONT`

Specifies to order the current rendering context in front of all the other contexts.

Available in Mac OS X v10.1 and later.

Declared in `agl.h`.

`AGL_CONTEXT_SURFACE_ID`

The associated value is the ID of the drawable surface for the rendering context. You can't set this value because the system sets it. However, you can retrieve the value using the function `aglGetInteger` (page 29).

Available in Mac OS X v10.2 and later.

Declared in `agl.h`.

`AGL_CONTEXT_DISPLAY_ID`

The associated value is a list of the display IDs of all displays touched by the rendering context, up to a maximum of 32 displays. You can't set this list of values because the system assigns display ID values. However, you can retrieve the value using the function `aglGetInteger` (page 29).

Available in Mac OS X v10.2 and later.

Declared in `agl.h`.

`AGL_SURFACE_ORDER`

The associated value is the position of the OpenGL surface relative to the window. A value of `1` means that the position is above the window; a value of `-1` specifies a position that is below the window.

Available in Mac OS X v10.2 and later.

Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_SURFACE_OPACITY`

The associated value specifies the opacity of the OpenGL surface. A value of `1` means the surface is opaque (the default); `0` means completely transparent.

Available in Mac OS X v10.2 and later.

Declared in `agl.h`.

`AGL_CLIP_REGION`

Enables or sets the drawable clipping region. The associated value is a `rgnHandle` data type that defines the clipping region.

Available in Mac OS X v10.2 and later.

Declared in `agl.h`.

`AGL_FS_CAPTURE_SINGLE`

Enables the capture of a single display for full-screen rendering. This option is disabled by default.

Available in Mac OS X v10.2 and later.

Declared in `agl.h`.

`AGL_SURFACE_BACKING_SIZE`

The associated value specifies the width and height of surface backing size.

Available in Mac OS X v10.4 and later.

Declared in `agl.h`.

`AGL_ENABLE_SURFACE_BACKING_SIZE`

Enable or disable the surface backing-size override.

Available in Mac OS X v10.4 and later.

Declared in `agl.h`.

`AGL_SURFACE_VOLATILE`

Flags the surface as a candidate for deletion.

Available in Mac OS X v10.4 and later.

Declared in `agl.h`.

Declared In

`agl.h`

Error Codes

Defines the error codes that can be returned by the `aglGetError` (page 28) function.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
#define AGL_NO_ERROR                0
#define AGL_BAD_ATTRIBUTE       10000
#define AGL_BAD_PROPERTY        10001
#define AGL_BAD_PIXELFMT        10002
#define AGL_BAD_RENDINFO        10003
#define AGL_BAD_CONTEXT         10004
#define AGL_BAD_DRAWABLE        10005
#define AGL_BAD_GDEV            10006
#define AGL_BAD_STATE           10007
#define AGL_BAD_VALUE           10008
#define AGL_BAD_MATCH           10009
#define AGL_BAD_ENUM            10010
#define AGL_BAD_OFFSCREEN       10011
#define AGL_BAD_FULLSCREEN      10012
#define AGL_BAD_WINDOW          10013
#define AGL_BAD_POINTER         10014
#define AGL_BAD_MODULE          10015
#define AGL_BAD_ALLOC           10016
#define AGL_BAD_CONNECTION      10017
```

Constants

`AGL_NO_ERROR`

No error.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_BAD_ATTRIBUTE`

Invalid pixel format attribute.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_BAD_PROPERTY`

Invalid renderer property.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_BAD_PIXELFMT`

Invalid pixel format.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_BAD_RENDINFO`

Invalid renderer info.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_BAD_CONTEXT`

Invalid rendering context.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_BAD_DRAWABLE`
: Invalid drawable object. A change of virtual screens (renderers) fails if the surface width and height cannot be set to the values requested, or if the creation of a drawable object or surfaces needed to support a pixel buffer fails for other reasons.

  Available in Mac OS X v10.0 and later.

  Declared in `agl.h`.

`AGL_BAD_GDEV`
: Invalid graphics device.

  Available in Mac OS X v10.0 and later.

  Declared in `agl.h`.

`AGL_BAD_STATE`
: Invalid rendering context state.

  Available in Mac OS X v10.0 and later.

  Declared in `agl.h`.

`AGL_BAD_VALUE`
: Invalid numerical value.

  Available in Mac OS X v10.0 and later.

  Declared in `agl.h`.

`AGL_BAD_MATCH`
: Invalid share rendering context.

  Available in Mac OS X v10.0 and later.

  Declared in `agl.h`.

`AGL_BAD_ENUM`
: Invalid enumerant.

  Available in Mac OS X v10.0 and later.

  Declared in `agl.h`.

`AGL_BAD_OFFSCREEN`
: Invalid offscreen drawable object.

  Available in Mac OS X v10.0 and later.

  Declared in `agl.h`.

`AGL_BAD_FULLSCREEN`
: Invalid full-screen drawable object.

  Available in Mac OS X v10.0 and later.

  Declared in `agl.h`.

`AGL_BAD_WINDOW`
: Invalid window.

  Available in Mac OS X v10.0 and later.

  Declared in `agl.h`.

`AGL_BAD_POINTER`
: Invalid pointer.

  Available in Mac OS X v10.0 and later.

  Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_BAD_MODULE`
Invalid code module.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_BAD_ALLOC`
Memory allocation failure.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_BAD_CONNECTION`
Unable to connect to the window server.

Available in Mac OS X v10.3 and later.

Declared in `agl.h`.

Discussion
Unlike many Carbon APIs, AGL functions don't return result codes for specific error conditions. Instead, AGL functions that fail either return `GL_FALSE` or `NULL`. You can find out the specific nature of the error by calling the function `aglGetError` (page 28), which returns the appropriate error code.

Declared In
`agl.h`

Globally Configured Options

Specify options that apply globally.

```
#define AGL_FORMAT_CACHE_SIZE 501
#define AGL_CLEAR_FORMAT_CACHE 502
#define AGL_RETAIN_RENDERERS 503
```

Constants
`AGL_FORMAT_CACHE_SIZE`
The associated value is the size of the positive pixel format cache and is initially set to `5`. If your application needs to use `n` different attribute lists to choose `n` different pixel formats repeatedly, then the application should set the cache size to `n` to maximize performance. After your application calls the `aglChoosePixelFormat` (page 11) function for the last time, you might want to set the cache size to `1` to minimize the memory used by the AGL library.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_CLEAR_FORMAT_CACHE`
When the associated value is nonzero, specifies to reset the pixel format cache. Clearing the cache does not affect the size of the cache for future storage of pixel formats. To minimize the memory consumed by the cache, your application should also set the cache size to `1`.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_RETAIN_RENDERERS`

Specifies whether or not to retain loaded plug-in renderers in memory. When the associated value is nonzero, the AGL library will not unload any plug-in renderers even if they are no longer in use. This option is useful to improve the performance of applications that repeatedly destroy and recreate their only (or last) rendering context. Normally, when the last rendering context created by a particular plug-in renderer is destroyed, that renderer is unloaded from memory. When the associated value is zero, AGL returns to its normal mode of operation and all renderers that are not in use are unloaded.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

Discussion

You can pass any of these options, along with the appropriate value, to the function `aglConfigure` (page 13).

Declared In

`agl.h`

Renderer Attributes

Define options for managing renderers.

```
#define AGL_RENDERER_ID 70
#define AGL_SINGLE_RENDERER 71
#define AGL_NO_RECOVERY 72
#define AGL_ACCELERATED 73
#define AGL_CLOSEST_POLICY 74
#define AGL_ROBUST 75
#define AGL_BACKING_STORE 76
#define AGL_MP_SAFE 78
#define AGL_WINDOW 80
#define AGL_MULTISCREEN 81
#define AGL_VIRTUAL_SCREEN 82
#define AGL_COMPLIANT 83
#define AGL_PBUFFER 90
#define AGL_REMOTE_PBUFFER 91
```

Constants

`AGL_RENDERER_ID`

The associated value is a nonnegative integer that specifies a renderer. If present, OpenGL renderers that match the specified ID are preferred. See "Renderer IDs" (page 72) for possible values. You can pass this constant to the function aglDescribeRenderer (page 18).

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_SINGLE_RENDERER`

This constant is a Boolean attribute. If it is present in the attributes array, specifies that a single pixel format represents a single renderer for all screens. On systems with multiple screens, this option disables the ability of the AGL library to drive different monitors through different graphics accelerator cards with a single AGL rendering context.. Do not supply a value with this constant because its presence in the array implies `true`.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_NO_RECOVERY`

This constant is a Boolean attribute. If it is present in the attributes array, specifies that failure recovery features are disabled for this pixel format. Normally, if an accelerated renderer cannot attach to a drawable object due to insufficient video memory, AGL automatically switches to another renderer. This attribute disables these features so that rendering will always be done by the chosen renderer. For example, with this option enabled, you won't get a software renderer as a fallback if a hardware-accelerated renderer runs out of resources.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_ACCELERATED`

This constant is a Boolean attribute. If it is present in the attributes array, specifies renderers that are attached to a hardware accelerated graphics device. It is usually impossible to support more than one accelerated graphics device, because typically when a window spans more than one device, OpenGL uses the software renderer. You can pass this constant to the function aglDescribeRenderer (page 18) to find out whether a particular renderer is hardware accelerated.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_CLOSEST_POLICY`

The associated value specifies a color buffer size. When set, this option alters the pixel format choosing policy such that a color buffer closest to the requested size is preferred, regardless of the actual color buffer depth of the supported graphics device.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_ROBUST`

This constant is a Boolean attribute. If it is present in the attributes array, specifies that AGL should consider only those renderers that do not have any failure modes associated with a lack of video card resources. You can pass this constant to the function aglDescribeRenderer (page 18).

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_BACKING_STORE`

This constant is a Boolean attribute. If it is present in the attributes array, specifies that AGL should consider only those renderers that have a back color buffer the full size of the drawable object (regardless of window visibility). AGL guarantees that the back buffer contents will be valid after a call to the `aglSwapBuffers` (page 44) function. You can pass this constant to the function aglDescribeRenderer (page 18).

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_MP_SAFE`

This constant is a Boolean attribute. If it is present in the attributes array, specifies a renderer that is multiprocessor safe. This attribute is deprecated in Mac OS X because all renderers can accept commands for threads running on a second processor. However, this does not mean that all renderers are thread-safe or reentrant.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_WINDOW`

This constant is a Boolean attribute. If it is present in the attributes array, specifies that the pixel format can be used to render to an onscreen window. You can pass this constant to the function aglDescribeRenderer (page 18).

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_MULTISCREEN`

This constant is a Boolean attribute. If it is present in the attributes array, specifies that the renderer is capable of driving multiple screens with the same rendering context. (A single window can span multiple screens.) You can pass this constant to the function aglDescribeRenderer (page 18).

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_VIRTUAL_SCREEN`

The associated value is an integer that specifies the virtual screen number of the pixel format.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_COMPLIANT`

This constant is a Boolean attribute. If it is present in the attributes array, specifies a pixel format that is fully compliant with OpenGL. All Mac OS X renderers are OpenGL-compliant. You can pass this constant to the function aglDescribeRenderer (page 18).

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_PBUFFER`

This constant is a Boolean attribute. If it is present in the attributes array, specifies that the renderer can render to a pixel buffer. You can pass this constant to the function aglDescribeRenderer (page 18).

Available in Mac OS X v10.3 and later.

Declared in `agl.h`.

`AGL_REMOTE_PBUFFER`

This constant is a Boolean attribute. If it is present in the attributes array, specifies that the renderer can render offline to a pixel buffer.

Available in Mac OS X v10.3 and later.

Declared in `agl.h`.

Declared In
`agl.h`

Renderer IDs

Define constants that specify hardware and software renderers.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
#define AGL_RENDERER_GENERIC_ID               0x00020200
#define AGL_RENDERER_GENERIC_FLOAT_ID         0x00020400
#define AGL_RENDERER_APPLE_SW_ID              0x00020600
#define AGL_RENDERER_ATI_RAGE_128_ID          0x00021000
#define AGL_RENDERER_ATI_RADEON_ID            0x00021200
#define AGL_RENDERER_ATI_RAGE_PRO_ID          0x00021400
#define AGL_RENDERER_ATI_RADEON_8500_ID       0x00021600
#define AGL_RENDERER_ATI_RADEON_9700_ID       0x00021800
#define AGL_RENDERER_ATI_RADEON_X1000_ID      0x00021900
#define AGL_RENDERER_NVIDIA_GEFORCE_2MX_ID    0x00022000
#define AGL_RENDERER_NVIDIA_GEFORCE_3_ID      0x00022200
#define AGL_RENDERER_NVIDIA_GEFORCE_FX_ID     0x00022400
#define AGL_RENDERER_VT_BLADE_XP2_ID          0x00023000
#define AGL_RENDERER_INTEL_900_ID             0x00024000
#define AGL_RENDERER_MESA_3DFX_ID             0x00040000
```

Constants

`AGL_RENDERER_GENERIC_ID`

A generic display device. (**Deprecated.** Deprecated on Intel-based Macintosh computers.)

Available in Mac OS X v10.0 and later.

Declared in `aglRenderers.h`.

`AGL_RENDERER_GENERIC_FLOAT_ID`

A floating-point software renderer that is optimized for vector-based processors, is programmable, and supports shading.

Available in Mac OS X v10.3 and later.

Declared in `aglRenderers.h`.

`AGL_RENDERER_APPLE_SW_ID`

The Apple software renderer.

Available in Mac OS X v10.2 and later.

Declared in `aglRenderers.h`.

`AGL_RENDERER_ATI_RAGE_128_ID`

An ATI Rage 128 display device.

Available in Mac OS X v10.2 and later.

Declared in `aglRenderers.h`.

`AGL_RENDERER_ATI_RADEON_ID`

An ATI Radeon display device.

Available in Mac OS X v10.2 and later.

Declared in `aglRenderers.h`.

`AGL_RENDERER_ATI_RAGE_PRO_ID`

An ATI Rage Pro display device.

Available in Mac OS X v10.2 and later.

Declared in `aglRenderers.h`.

`AGL_RENDERER_ATI_RADEON_8500_ID`

An ATI Radeon 8500 display device.

Available in Mac OS X v10.2 and later.

Declared in `aglRenderers.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_RENDERER_ATI_RADEON_9700_ID`
An ATI Radeon 9700 display device.

Available in Mac OS X v10.3 and later.

Declared in `aglRenderers.h`.

`AGL_RENDERER_ATI_RADEON_X1000_ID`
An ATI Radeon 9700 display device.

Available in Mac OS X v10.5 and later.

Declared in `aglRenderers.h`.

`AGL_RENDERER_NVIDIA_GEFORCE_2MX_ID`
An NVIDIA GeForce 2MX display device or an NVIDIA GeForce 4MX.

Available in Mac OS X v10.2 and later.

Declared in `aglRenderers.h`.

`AGL_RENDERER_NVIDIA_GEFORCE_3_ID`
An NVIDIA GeForce 3 display device or an NVIDIA GeForce 4 Ti.

Available in Mac OS X v10.2 and later.

Declared in `aglRenderers.h`.

`AGL_RENDERER_NVIDIA_GEFORCE_FX_ID`
An NVIDIA GeForce FX, GeForce 6, or GeForce 7 display device.

Available in Mac OS X v10.3 and later.

Declared in `aglRenderers.h`.

`AGL_RENDERER_VT_BLADE_XP2_ID`
A Village Tronic display device.

Available in Mac OS X v10.3 and later.

Declared in `aglRenderers.h`.

`AGL_RENDERER_INTEL_900_ID`
An Intel GMA 900 display device.

Available in Mac OS X v10.5 and later.

Declared in `aglRenderers.h`.

`AGL_RENDERER_MESA_3DFX_ID`
A Mesa 3DFX display device.

Available in Mac OS X v10.2 and later.

Declared in `aglRenderers.h`.

Declared In
`aglrenderers.h`

Renderer Properties

Specify constants that you can use to query renderer properties.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
#define AGL_BUFFER_MODES 100
#define AGL_MIN_LEVEL 101
#define AGL_MAX_LEVEL 102
#define AGL_COLOR_MODES 103
#define AGL_ACCUM_MODES 104
#define AGL_DEPTH_MODES 105
#define AGL_STENCIL_MODES 106
#define AGL_MAX_AUX_BUFFERS 107
#define AGL_VIDEO_MEMORY 120
#define AGL_TEXTURE_MEMORY 121
#define AGL_RENDERER_COUNT 128
```

Constants

`AGL_BUFFER_MODES`

The associated value can be a bitwise `OR` of the any of the constants specified in "Buffer Mode Flags" (page 57).

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_MIN_LEVEL`

The associated value specifies the minimum overlay buffer level. Negative values indicate an underlay buffer.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_MAX_LEVEL`

The associated value specifies the maximum overlay buffer level.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_COLOR_MODES`

The associated value can be a bitwise `OR` of any of the constants specified in "Color Modes" (page 58).

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_ACCUM_MODES`

The associated value can be a bitwise `OR` of any of the constants specified in "Color Modes" (page 58).

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_DEPTH_MODES`

The associated value can be the bitwise `OR` of any of the constants specified in "Bit Depths" (page 50).

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_STENCIL_MODES`

The associated value can be the bitwise `OR` of any of the constants specified in "Bit Depths" (page 50).

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

`AGL_MAX_AUX_BUFFERS`

The associated value is the maximum number of auxiliary buffers that can be supported by the renderer.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_VIDEO_MEMORY`

The associated value is the amount of video memory.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_TEXTURE_MEMORY`

The associated value is the amount of texture memory.

Available in Mac OS X v10.0 and later.

Declared in `agl.h`.

`AGL_RENDERER_COUNT`

The associated value is the number of renderers.

Available in Mac OS X v10.3 and later.

Declared in `agl.h`.

Discussion

You can pass these constants to the function aglDescribeRenderer (page 18) to find out the property value for a specific renderer.

Declared In

`agl.h`

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Document Revision History

This table describes the changes to *AGL Reference*.

Date	Notes
2007-10-31	Updated with three new functions.
	Added the functions `aglCreatePixelFormat` (page 16) and `aglDisplaysOfPixelFormat` (page 23).
	Added deprecation summaries to `aglDevicesOfPixelFormat` (page 22) and `aglQueryRendererInfo` (page 34).
	Corrected a typographical error in the function name `aglQueryRendererInfoForCGDirectDisplayIDs` (page 35).
2007-07-18	Updated for Mac OS X v10.5.
	Added the functions `aglSetWindowRef` (page 42), `aglGetWindowRef` (page 31), `aglSetHIViewRef` (page 38), `aglGetHIViewRef` (page 28), and `aglQueryRendererInfoForCGDirectDisplayIDs` (page 35).
	Updated `aglChoosePixelFormat` (page 11) and "`Buffer and Renderer Attributes`" (page 52).
	Revised the texture target information in `aglCreatePBuffer` (page 15).
	Revised the discussion of `aglSetFullScreen` (page 37).
	Revised the discussion of `aglSetPBuffer` (page 40).
	Updated renderers in "`Renderer IDs`" (page 72).
2006-05-23	Made content and formatting changes throughout the book.
	Revised "Introduction" (page 7).
	Edited content to make it more consistent with other Apple OpenGL documentation.
	Fixed formatting.
	Added See Also sections.
2005-12-06	Updated several function descriptions.
2005-10-04	Updated the introduction. Made numerous editorial and stylistic changes throughout.

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Date	Notes
	Revised the Introduction to include a definition of the term drawable object, which is now used throughout the reference.
	Removed conceptual and task information from the Introduction because this information is in AGL Programming Guide.
	Organized functions by task.
	Added availability and header information.
	Added reference documentation for symbols in the `aglRenderers.h` header file.
	Removed duplicate material from the Constants section.
	Fixed typographical errors.
2005-08-11	Changed title from "Apple OpenGL Reference."
2005-04-29	Updated for Mac OS X 10.4 Tiger

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

Index

A

aglChoosePixelFormat function 11
aglConfigure function 13
AGLContext data type 48
aglCopyContext function 13
aglCreateContext function 14
aglCreatePBuffer function 15
aglCreatePixelFormat function 16
aglDescribePBuffer function 17
aglDescribePixelFormat function 18
aglDescribeRenderer function 18
aglDestroyContext function 19
aglDestroyPBuffer function 20
aglDestroyPixelFormat function 21
aglDestroyRendererInfo function 21
AGLDevice data type 48
aglDevicesOfPixelFormat function 22
aglDisable function 23
aglDisplaysOfPixelFormat function 23
AGLDrawable data type 48
aglEnable function 24
aglErrorString function 25
aglGetCGLContext function 25
aglGetCGLPixelFormat function 26
aglGetCurrentContext function 27
aglGetDrawable function 27
aglGetError function 28
aglGetHIViewRef function 28
aglGetInteger function 29
aglGetPBuffer function 29
aglGetVersion function 30
aglGetVirtualScreen function 31
aglGetWindowRef function 31
aglIsEnabled function 32
aglNextPixelFormat function 32
aglNextRendererInfo function 33
AGLPbuffer data type 49
AGLPixelFormat data type 49
aglQueryRendererInfo function 34
aglQueryRendererInfoForCGDirectDisplayIDs function 35
AGLRendererInfo data type 50
aglResetLibrary function 35
aglSetCurrentContext function 35
aglSetDrawable function 36
aglSetFullScreen function 37
aglSetHIViewRef function 38
aglSetInteger function 38
aglSetOffScreen function 39
aglSetPBuffer function 40
aglSetVirtualScreen function 41
aglSetWindowRef function 42
aglSurfaceTexture function 43
aglSwapBuffers function 44
aglTexImagePBuffer function 44
aglUpdateContext function 46
aglUseFont function 46
AGL_0_BIT constant 50
AGL_10_BIT constant 51
AGL_128_BIT constant 52
AGL_12_BIT constant 51
AGL_16_BIT constant 51
AGL_1_BIT constant 50
AGL_24_BIT constant 51
AGL_2_BIT constant 51
AGL_32_BIT constant 52
AGL_3_BIT constant 51
AGL_48_BIT constant 52
AGL_4_BIT constant 51
AGL_5_BIT constant 51
AGL_64_BIT constant 52
AGL_6_BIT constant 51
AGL_8_BIT constant 51
AGL_96_BIT constant 52
AGL_ACCELERATED constant 71
AGL_ACCUM_ALPHA_SIZE constant 55
AGL_ACCUM_BLUE_SIZE constant 55
AGL_ACCUM_GREEN_SIZE constant 55
AGL_ACCUM_MODES constant 75
AGL_ACCUM_RED_SIZE constant 55
AGL_ALL_RENDERERS constant 53

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

AGL_ALPHA_SIZE constant 55
AGL_ARGB12121212_BIT constant 62
AGL_ARGB1555_BIT constant 60
AGL_ARGB16161616_BIT constant 62
AGL_ARGB2101010_BIT constant 61
AGL_ARGB4444_BIT constant 60
AGL_ARGB8888_BIT constant 61
AGL_AUX_BUFFERS constant 54
AGL_AUX_DEPTH_STENCIL constant 57
AGL_BACKING_STORE constant 71
AGL_BAD_ALLOC constant 69
AGL_BAD_ATTRIBUTE constant 67
AGL_BAD_CONNECTION constant 69
AGL_BAD_CONTEXT constant 67
AGL_BAD_DRAWABLE constant 68
AGL_BAD_ENUM constant 68
AGL_BAD_FULLSCREEN constant 68
AGL_BAD_GDEV constant 68
AGL_BAD_MATCH constant 68
AGL_BAD_MODULE constant 69
AGL_BAD_OFFSCREEN constant 68
AGL_BAD_PIXELFMT constant 67
AGL_BAD_POINTER constant 68
AGL_BAD_PROPERTY constant 67
AGL_BAD_RENDINFO constant 67
AGL_BAD_STATE constant 68
AGL_BAD_VALUE constant 68
AGL_BAD_WINDOW constant 68
AGL_BGR233_A8_BIT constant 60
AGL_BGR233_BIT constant 59
AGL_BLUE_SIZE constant 55
AGL_BUFFER_MODES constant 75
AGL_BUFFER_NAME constant 65
AGL_BUFFER_RECT constant 64
AGL_BUFFER_SIZE constant 53
AGL_CLEAR_FORMAT_CACHE constant 69
AGL_CLIP_REGION constant 66
AGL_CLOSEST_POLICY constant 71
AGL_COLORMAP_ENTRY constant 64
AGL_COLORMAP_TRACKING constant 64
AGL_COLOR_FLOAT constant 57
AGL_COLOR_MODES constant 75
AGL_COMPLIANT constant 72
AGL_CONTEXT_DISPLAY_ID constant 65
AGL_CONTEXT_SURFACE_ID constant 65
AGL_DEPTH_MODES constant 75
AGL_DEPTH_SIZE constant 55
AGL_DOUBLEBUFFER constant 54
AGL_DOUBLEBUFFER_BIT constant 58
AGL_ENABLE_SURFACE_BACKING_SIZE constant 66
AGL_FORMAT_CACHE_SIZE constant 69
AGL_FS_CAPTURE_SINGLE constant 66
AGL_FULLSCREEN constant 56
AGL_GREEN_SIZE constant 54
AGL_INDEX16_BIT constant 62
AGL_INDEX8_BIT constant 62
AGL_LEVEL constant 54
AGL_MAXIMUM_POLICY constant 56
AGL_MAX_AUX_BUFFERS constant 76
AGL_MAX_LEVEL constant 75
AGL_MINIMUM_POLICY constant 56
AGL_MIN_LEVEL constant 75
AGL_MONOSCOPIC_BIT constant 58
AGL_MP_SAFE constant 71
AGL_MULTISAMPLE constant 57
AGL_MULTISCREEN constant 72
AGL_NONE constant 53
AGL_NO_ERROR constant 67
AGL_NO_RECOVERY constant 71
AGL_OFFSCREEN constant 56
AGL_ORDER_CONTEXT_TO_FRONT constant 65
AGL_PBUFFER constant 72
AGL_PIXEL_SIZE constant 56
AGL_RASTERIZATION constant 65
AGL_RED_SIZE constant 54
AGL_REMOTE_PBUFFER constant 72
AGL_RENDERER_APPLE_SW_ID constant 73
AGL_RENDERER_ATI_RADEON_8500_ID constant 73
AGL_RENDERER_ATI_RADEON_9700_ID constant 74
AGL_RENDERER_ATI_RADEON_ID constant 73
AGL_RENDERER_ATI_RADEON_X1000_ID constant 74
AGL_RENDERER_ATI_RAGE_128_ID constant 73
AGL_RENDERER_ATI_RAGE_PRO_ID constant 73
AGL_RENDERER_COUNT constant 76
AGL_RENDERER_GENERIC_FLOAT_ID constant 73
AGL_RENDERER_GENERIC_ID constant 73
AGL_RENDERER_ID constant 70
AGL_RENDERER_INTEL_900_ID constant 74
AGL_RENDERER_MESA_3DFX_ID constant 74
AGL_RENDERER_NVIDIA_GEFORCE_2MX_ID constant 74
AGL_RENDERER_NVIDIA_GEFORCE_3_ID constant 74
AGL_RENDERER_NVIDIA_GEFORCE_FX_ID constant 74
AGL_RENDERER_VT_BLADE_XP2_ID constant 74
AGL_RETAIN_RENDERERS constant 70
AGL_RGB101010_A8_BIT constant 62
AGL_RGB101010_BIT constant 61
AGL_RGB121212_BIT constant 62
AGL_RGB161616_BIT constant 62
AGL_RGB332_A8_BIT constant 60
AGL_RGB332_BIT constant 60
AGL_RGB444_A8_BIT constant 60
AGL_RGB444_BIT constant 60
AGL_RGB555_A8_BIT constant 61
AGL_RGB555_BIT constant 60
AGL_RGB565_A8_BIT constant 61
AGL_RGB565_BIT constant 61

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.

AGL_RGB888_A8_BIT constant 61
AGL_RGB888_BIT constant 61
AGL_RGB8_A8_BIT constant 59
AGL_RGB8_BIT constant 59
AGL_RGBA constant 54
AGL_RGBAFLOAT128_BIT constant 63
AGL_RGBAFLOAT256_BIT constant 63
AGL_RGBAFLOAT64_BIT constant 62
AGL_RGBFLOAT128_BIT constant 63
AGL_RGBFLOAT256_BIT constant 63
AGL_RGBFLOAT64_BIT constant 62
AGL_ROBUST constant 71
AGL_SAMPLES_ARB constant 57
AGL_SAMPLE_ALPHA constant 57
AGL_SAMPLE_BUFFERS_ARB constant 56
AGL_SINGLEBUFFER_BIT constant 58
AGL_SINGLE_RENDERER constant 70
AGL_STATE_VALIDATION constant 65
AGL_STENCIL_MODES constant 75
AGL_STENCIL_SIZE constant 55
AGL_STEREO constant 54
AGL_STEREOSCOPIC_BIT constant 58
AGL_SUPERSAMPLE constant 57
AGL_SURFACE_BACKING_SIZE constant 66
AGL_SURFACE_OPACITY constant 66
AGL_SURFACE_ORDER constant 65
AGL_SURFACE_VOLATILE constant 66
AGL_SWAP_INTERVAL constant 65
AGL_SWAP_LIMIT constant 64
AGL_SWAP_RECT constant 64
AGL_TEXTURE_MEMORY constant 76
AGL_VIDEO_MEMORY constant 76
AGL_VIRTUAL_SCREEN constant 72
AGL_WINDOW constant 72

B

Bit Depths 50
Buffer and Renderer Attributes 52
Buffer Mode Flags 57

C

Color Modes 58
Context Options and Parameters 63

E

Error Codes 66

G

Globally Configured Options 69

R

Renderer Attributes 70
Renderer IDs 72
Renderer Properties 74

2007-10-31 | © 2004, 2007 Apple Inc. All Rights Reserved.