AGL Reference

For OpenGL for Mac OS X

Apple Computer, Inc.
Technical Publications
July, 2002

About This Book

This reference book is designed and written for Mac OS developers who are working with or building applications using OpenGL. This book provides a reference to the AGL library, which provides features of OpenGL that, by their nature, must be operating system specific. This release of the software corresponds to version 1.3 of OpenGL from Silicon Graphics, Inc. (SGI).

This book is intended for use in conjunction with the following documents:

- *OpenGL Introduction*, which provides an overview of OpenGL and Apple's implementation of it. This document is included as part of the OpenGL for Mac OS X CFM Software Developer Kit (SDK).
- *OpenGL 1.3 Specification*, which is the official specification for OpenGL. This document is included as part of the OpenGL for Mac OS X CFM Software Developer Kit (SDK) and at www.opengl.org.
- *OpenGL State Machine*, which illustrates the OpenGL state machine and graphics pipeline. This document is included as part of the OpenGL for Mac OS X CFM Software Developer Kit (SDK) and at www.opengl.org.
- *OpenGL Reference*, which describes GL, the main OpenGL library. This document is available at www.opengl.org.
- *OpenGL GLU Reference*, which describes the OpenGL Utility Library, containing graphical extensions based entirely on GL functions. This document is available at www.opengl.org.
- *OpenGL GLUT Reference*, which describes the OpenGL Utility Toolkit, a standard API for performing operations associated with a windowing environment. This document is available at www.opengl.org.

Conventions Used in This Book

This book provides various conventions to present information. Words that require special treatment appear in specific fonts or font styles. Certain types of information, such as parameter blocks, use special fonts so that you can scan them quickly.

Special Fonts

All code listings, reserved words, and the names of actual data structures, constants, fields, parameters, and functions are shown in Courier New `(this is Courier New).`

Words that appear in **boldface** are cross-references to key terms or concepts that are defined elsewhere in the manual.

Types of Notes

There are several types of notes used in this book.

Note
A note like this contains information that is interesting but not essential to an understanding of the main text. ◆

IMPORTANT
A note like this contains information that is essential for an understanding of the main text. ▲

WARNING
A warning like this indicates potential problems that you should be aware of as you design your software. Failure to heed these warnings could result in system crashes or loss of data. ▲

Development Environment

OpenGL for Macintosh is implemented as a set of Frameworks in Mac OS X. To provide support for both Mach-O and CFM based binaries, a bridge library is also provided. See *Inside Mac OS X: System Overview* for more on Mach-O and CFM binary formats. Both Project Builder and Metrowerks CodeWarrior can be used to develop AGL based OpenGL code.

Code listings in this book are shown in ANSI C. They suggest methods of using various functions and illustrate techniques for accomplishing particular tasks. Although most code listings have been compiled and tested, Apple Computer Inc., does not intend for you to use these code samples unmodified or untested in your application.

System Requirements

OpenGL is supported on systems, which are officially supported by Mac OS X. The specific features may vary based on the capabilities of renderers and what set of features each offer.

Software Development Kit (SDK)

Mac OS X Developer CDs provide all the required Frameworks (including headers and link libraries) for producing AGL based OpenGL apps. The Mac OS X OpenGL CFM SDK provides additional resources for developers working with Carbon CFM based code. It is available for download from developer.apple.com/opengl/ and provided on CD-ROM in the Apple Developer Connection monthly mailing program (see www.apple.com/developer/programs/ for membership information).

AGL Introduction

In Mac OS X, features of OpenGL, which are specific to the Mac OS, are implemented by one of three libraries, Core OpenGL (CGL), Cocoa OpenGL (NSGL) and Carbon OpenGL (AGL). AGL should be used for applications that need to run on versions of the Mac OS prior to Mac OS X and for Carbon applications in general. Full screen Carbon applications can use AGL or CGL Mac OS X specific functions.

This book documents the AGL library, which is implemented as a Mac OS X Framework and a CFM bridge library. First, there is this short overview, which is followed by a section presenting all AGL functions in reference page format.

AGL extends the capabilities of a Mac OS window with several buffers other than the standard color buffer. These buffers include back and auxiliary color buffers, depth buffers, a stencil buffer, and a color accumulation buffer.

Two of the complex data types used by the AGL API are derived from standard Carbon data types; the `AGLDrawable` type corresponds to a Carbon `CGrafPtr` and the `AGLDevice` type corresponds to a Carbon `GDHandle`.

To render using OpenGL into a Mac OS graphics port, or drawable, you first must choose a pixel format that defines the required OpenGL buffers using **`aglChoosePixelFormat`** to select a compatible pixel format. Note **`aglChoosePixelFormat`** is one of four functions, which initialize the library and must be called prior to any other AGL function. The other three functions are **`aglQueryRendererInfo`**, **`aglCreateContext`**, and **`aglSetVirtualScreen`**.

Use the selected pixel format to create an AGL context. AGL contexts are created with **`aglCreateContext`**. Then create a Mac OS drawable by using **`CreateNewWindow`**, or another related Carbon function, and retrieve the `AGLDrawable` (e.g., `CGrafPtr)` associated with the window usually with **`GetWindowPort`**. Finally, bind the context and the drawable together using **`aglSetDrawable`** and make the context the current context with **`aglSetCurrentContext`**. You must use **`GetWindowPort`** to retrieve the `CGrafPtr` from the `WindowRef` as you cannot simply cast opaque data references. This context/drawable pair becomes the current context and current drawable, and it is used by all OpenGL commands until **`aglSetCurrentContext`** or **`aglSetDrawable`** is called with a different argument.

Listing 1-1 shows the minimum code required to create a Mac OS window compatible with OpenGL, in RGBA-format, and clear it yellow. The code is correct and Carbon compatible, but it does not include any error checking.

Listing 1-1

```
#include <Carbon.h>
#include <agl.h>

int main(void)
{
    Rect rect;
    WindowRef win = NULL;
    GLint attrib [32], i = 0;
    AGLPixelFormat fmt;
    AGLContext ctx;
    unsigned long temp;

    /* Initialize Mac OS */
    InitCursor();

    /* Create a window */
    SetRect(&rect, 50, 50, 450, 450);
    CreateNewWindow (kDocumentWindowClass,
        kWindowStandardDocumentAttributes, &rect, &win);
    ShowWindow (win);

    /* Choose pixel format */
    attrib [i++] = AGL_RGBA;
    attrib [i++] = AGL_DOUBLEBUFFER;
    attrib [i++] = AGL_NONE; // end parameter list
    fmt = aglChoosePixelFormat(NULL, 0, attrib);

    /* Create an AGL context */
    ctx = aglCreateContext (fmt, NULL);

    /* Attach the context to the window  and set context current */
    aglSetDrawable (ctx, GetWindowPort (win));
    aglSetCurrentContext (ctx);

    /* Clear buffer */
    glClearColor (1.0, 1.0, 0.0, 1.0);
    glClear (GL_COLOR_BUFFER_BIT);

    /* swap to front */
    aglSwapBuffers (ctx);

    Delay (60, &temp);

    return 0;
}
```

Note that the application must create an AGL context and attach it to a drawable before OpenGL commands can be executed. OpenGL commands issued while no context/drawable pair is current are ignored. AGL functions have undefined behavior prior to the library being initialized, which occurs when **aglChoosePixelFormat**, **aglQueryRendererInfo**, **aglCreateContext**, or **aglSetVirtualScreen** is called.

AGL Reference

This section documents all AGL commands. Each command is presented in reference page format.

aglChoosePixelFormat

Select a pixel format to match specified attributes

C SPECIFICATION

```
#include <agl.h>
AGLPixelFormat aglChoosePixelFormat (AGLDevice *gdev,
    GLint ndev,
    const GLint *attribs )
```

PARAMETERS

gdev	An array of Mac OS graphics devices (type `GDHandle`)
ndev	The number of graphics devices in *gdev*
attribs	Specifies a list of Boolean attributes and integer attribute/value pairs. The last attribute must be `AGL_NONE`.

DESCRIPTION

`aglChoosePixelFormat` returns a pointer to data describing a pixel format that is supported by all the graphics devices in *gdev* and best meets the specification defined by *attribs* . If *gdev* and *ndev* are set to `NULL` and zero, respectively, **`aglChoosePixelFormat`** will return a pixel format that is supported by all graphics devices on the system.

The Boolean AGL attributes of the returned format will match the specified values, and the integer AGL attributes will be as close to the specified values as can be provided by the system. If no conforming pixel format exists, `NULL` is returned. To free the data returned by this function, use **`aglDestroyPixelFormat`**. The **`AGL_MINIMUM_POLICY`** and **`AGL_MAXIMUM_POLICY`** attributes can be used to alter the selection criteria.

All Boolean AGL attributes default to **`GL_FALSE`**. All integer AGL attributes default to zero. Default specifications are superseded by attributes included in *attribs*. Boolean attributes included in *attribs* are understood to be **`GL_TRUE`**. Integer attributes are followed immediately by the corresponding desired value. The list must be terminated with **`AGL_NONE`**.

The interpretations of the AGL pixel format attributes are as follows:

`AGL_BUFFER_SIZE`
Must be followed by a nonnegative integer that indicates the desired color index buffer size. The smallest color index buffer of at least the specified size is preferred. Ignored if **`AGL_RGBA`** is asserted. Note, no current Mac OS X renderers support color index mode.

`AGL_LEVEL`
Must be followed by an integer buffer-level specification. This specification is honored exactly. 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. No current

Mac OS X renderers support overlay or underlay fame buffers.

`AGL_RGBA`
If present, only RGBA pixel formats are considered. Otherwise, only color index pixel formats are considered. In most cases, applications will specify this attribute.

`AGL_DOUBLEBUFFER`
If present, only double-buffered pixel formats are considered. Otherwise, only single-buffered pixel formats are considered. For most applications, which are not specifically looking to be single buffered, this attribute will be specified.

`AGL_STEREO`
If present, only stereo pixel formats are considered. Otherwise, only monoscopic pixel formats are considered.

`AGL_AUX_BUFFERS`
Must be followed by a nonnegative integer that indicates the desired number of auxiliary buffers. Pixel formats with the smallest number of auxiliary buffers that meets or exceeds the specified number are preferred.

`AGL_RED_SIZE`
Must be followed by a nonnegative buffer size specification. A red buffer that most closely matches the specified size is preferred. Mac OS X currently only supports 5 bit or 8 bit color components.

`AGL_GREEN_SIZE`
Must be followed by a nonnegative buffer size specification. A green buffer that most closely matches the specified size is preferred. Mac OS X currently only supports 5 bit or 8 bit color components.

`AGL_BLUE_SIZE`
Must be followed by a nonnegative buffer size specification. A blue buffer that most closely matches the specified size is preferred. Mac OS X currently only supports 5 bit or 8 bit color components.

`AGL_ALPHA_SIZE`
Must be followed by a nonnegative buffer size specification. An alpha buffer that most closely matches the specified size is preferred. Mac OS X currently supports two pixel formats with alpha, 32 bit 8888 ARGB and 16 bit 1555 ARGB.

`AGL_DEPTH_SIZE`
Must be followed by a nonnegative depth buffer size specification. A depth buffer that most closely matches the specified size is preferred. Applications that want to a z-buffer must specify this attribute.

`AGL_STENCIL_SIZE`
Must be followed by a nonnegative integer that indicates the desired number of stencil bitplanes. The smallest stencil buffer of at least the specified size is preferred.

`AGL_ACCUM_RED_SIZE`
Must be followed by a nonnegative buffer size specification. A red accumulation buffer that most closely matches the specified size is preferred. In Mac OS X, as of this writing, this attribute, if supported, will always result in an accumulation buffer with either 8 or 16 bits per color component.

`AGL_ACCUM_GREEN_SIZE`
Must be followed by a nonnegative buffer size specification. A green accumulation buffer that most closely matches the specified size is preferred. In Mac OS X, as of this writing, this attribute, if supported, will

always result in an accumulation buffer with either 8 or 16 bits per color component.

AGL_ACCUM_BLUE_SIZE
Must be followed by a nonnegative buffer size specification. A blue accumulation buffer that most closely matches the specified size is preferred. In Mac OS X, as of this writing, this attribute, if supported, will always result in an accumulation buffer with either 8 or 16 bits per color component.

AGL_ACCUM_ALPHA_SIZE
Must be followed by a nonnegative buffer size specification. An alpha accumulation buffer that most closely matches the specified size is preferred. In Mac OS X, as of this writing, this attribute, if supported, will always result in an accumulation buffer with either 8 or 16 bits per component.

AGL_PIXEL_SIZE
Must be followed by a nonnegative bits-per-pixel specification that is matched exactly. 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, it's size is not included in the pixel size. Mac OS X currently supports 16 bit 1555 ARGB and 32 bit 8888 ARGB pixel sizes.

AGL_MINIMUM_POLICY
If present, 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.

AGL_MAXIMUM_POLICY
If present, 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.

AGL_CLOSEST_POLICY
If present, the pixel format choosing policy is altered for the color buffer such that the buffer closest to the requested size is preferred, regardless of the actual color buffer depth of the supported graphics device.

AGL_OFFSCREEN
If present, 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. Furthermore, *gdev* and *ndev* must be set to `NULL` and zero when **AGL_OFFSCREEN** is present. When **AGL_OFFSCREEN** is present, the **AGL_CLOSEST_POLICY** attribute is implied. No current Mac OS X hardware renderer supports accelerated off-screen rendering with this attribute. Accelerated off-screen rendering can be achieved simply by attaching a context to a drawable that is not currently visible (e.g., a window that is not shown).

AGL_FULLSCREEN
If present, only renderers that are capable of rendering to a full-screen graphics device are considered. Furthermore, *gdev* and *ndev* must be set to a valid **gDevice**, which will be the target device for rendering, when **AGL_FULLSCREEN** is present. After receiving a valid pixel format, an application can call **aglSetFullScreen** to switch resolutions and enable full screen rendering. Note, in this mode buffer swaps are likely made on hardware using page flipping.

AGL_ALL_RENDERERS
If present, pixel format selection will be open to all available renderers,

including debug and special-purpose renderers that are not OpenGL compliant.

`AGL_RENDERER_ID`
This attribute, must be followed by a non-negative renderer ID number. If present, OpenGL renderers that match the specified ID are preferred. Constants are provided in the "aglRenderers.h" header to select specific renderers. Of note is **`AGL_RENDERER_GENERIC_ID`**, which selects the Apple software renderer. The other constants select renderers for specific hardware vendors.

`AGL_SINGLE_RENDERER`
If present, a single rendering engine is chosen to render to all specified graphics devices. On systems with multiple screens, this disables the AGL library's ability to drive different monitors through different graphics accelerator cards with a single AGL context.

`AGL_NO_RECOVERY`
If present, the AGL library's failure recovery mechanisms are disabled, such that if an accelerated renderer fails to lack of resources, AGL will not automatically switch to another renderer. This condition normally would occur only when the actual set of frame buffers do not fit in VRAM after paging out all possible textures. This attribute disables this fall back feature forcing rendering to always be done by the chosen renderer. Without this attribute, pixel storage formats will be determined by the union of supported formats across all allocated renderers, which may result in less than optimal pixel packing.

`AGL_ACCELERATED`
If present, only renderers that are attached to a hardware accelerated graphics device are considered. It is possible to support more than one graphics device if the **`AGL_ACCELERATED`** attribute is given since Mac OS X will automatically switch renders for drawables that are moved from one device to another.

`AGL_BACKING_STORE`
If present, the only renderers considered are those that have a back color buffer the full size of the drawable (regardless of window visibility) and that guarantee the back buffer contents to be valid after a call to **`aglSwapBuffers`**. Normally windowed rendering context will have a backing store and full screen context will not due to different buffer swapping mechanics.

`AGL_ROBUST`
If present, only renderers that do not have any failure modes associated with a lack of video card resources are considered.

`AGL_MP_SAFE`
If present, only renderers that are multi-processor (MP) safe are considered Currently are renderers on OS X are MP Safe.

`AGL_SAMPLE_BUFFERS_ARB`
If present, must be followed a non-negative number of multi-sample buffers requested. The number of multi-sample buffers that most closely matches the specified size is preferred. This attribute should be used along with the **`AGL_SAMPLES_ARB`** attribute to enable full scene antialiasing. See the *GL_ARB_multisample* extension specification more information. Normally, this attribute will be set to 1 to enable multi-sampling. Other values will likely result in a single sample buffer being allocated. **(New in AGL for Max OS X 10.2 Jaguar)**

`AGL_SAMPLES_ARB`

If present, must be followed a non-negative number of samples per buffer requested. The number of samples per buffer that most closely matches the specified size is preferred. This attribute should be used along with the **`AGL_SAMPLE_BUFFERS_ARB`** attribute to enable full scene antialiasing. See the *GL_ARB_multisample* extension specification for more information. Normally, this value is 2 or more and indicates the number of samples per pixel in the frame, depth and stencil buffers (depending on which of these are enabled). . **(New in AGL for Max OS X 10.2 Jaguar)**

EXAMPLES

```
attribs  = {AGL_RGBA, AGL_DEPTH_SIZE, 16, AGL_NONE};
```

Specifies a single-buffered RGB pixel format in the normal frame buffer. The returned pixel format has color depth equal to the depth of the deepest graphics device on the system. It has a depth buffer as close to 16 bits as can be provided. It does not support double-buffering, or stereo display. It may or may not have one or more auxiliary color buffers, a stencil buffer, or an accumulation buffer.

NOTES

Avoid specifying pixel formats with an alpha color plane if no blending mode requiring the destination alpha value is used. This technique offers greater speed and may reduce memory usage.

If *gdev* specifies more than one graphics device (or is `NULL` on multi-screen system) **`aglChoosePixelFormat`** attempts to find a renderer or renderers to support all the devices with one AGL context. If a single hardware-accelerated renderer is found that can support the requested pixel format on all devices, this renderer is chosen. If accelerated renderers are found that can support only a subset of the devices, then pixel formats from multiple renderers are chosen. In this way, the hardware-accelerated renderer for the device on which the greatest number of the pixels of the drawable are located will be used at any given time, thus the current renderer can change as the drawable is resized and/or moved.

ERRORS

`aglChoosePixelFormat` returns `NULL` if it fails for any reason.

`AGL_BAD_ATTRIBUTE` is set if an invalid attribute is encountered in attribs.

`AGL_BAD_VALUE` is set if ndev is zero and gdev is not `NULL`.

`AGL_BAD_VALUE` is set if the **`AGL_OFFSCREEN`** attribute is specified and *gdev* is not `NULL`.

`AGL_BAD_GDEV` is set if *ndev* is nonzero and *gdev* is not a valid graphics device handle.

Other errors may be set by the OpenGL rendering engine.

SEE ALSO

`aglCreateContext`, **`aglDescribePixelFormat`**, **`aglDestroyPixelFormat`**

aglConfigure

Set the values of global configurable parameters

C SPECIFICATION

```
#include <agl.h>
GLboolean aglConfigure (  GLenum pname,
   GLuint param )
```

PARAMETERS

pname	Specifies the name of the parameter to be configured.
param	Specifies the new value of the parameter.

DESCRIPTION

Use **aglConfigure** to change the values of parameters that affect the operation of the AGL library. These parameter settings affect all contexts, not just the current context.

pname may take one of the following values:

AGL_FORMAT_CACHE_SIZE
param specifies the positive pixel format cache size. After an application has called **aglChoosePixelFormat** for the last time, it may set the cache size to one to minimize the memory used by the AGL library. If an application intends 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. The cache size is initially set to 5.

AGL_CLEAR_FORMAT_CACHE
If *param* is nonzero, the pixel format cache contents are freed. This does not affect the size of the cache for future storage of pixel formats. To minimize the memory consumed by the cache, the application should also set the cache size to 1.

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

ERRORS

aglConfigure returns **GL_FALSE** if it fails for any reason, **GL_TRUE** otherwise.

AGL_BAD_ENUM is set if either *pname* is not an accepted value.

AGL_BAD_VALUE is set if *param* is not an appropriate setting for *pname*.

SEE ALSO

`aglSetInteger`

aglCopyContext

Copy state from one rendering context to another.

C SPECIFICATION

```
#include <agl.h>
GLboolean aglCopyContext ( AGLContext src,
   AGLContext dst,
   GLuint mask )
```

PARAMETERS

src	Specifies the source context.
dst	Specifies the destination context.
mask	Specifies which portions of src state are to be copied to *dst* .

DESCRIPTION

aglCopyContext copies selected groups of state variables from *src* to *dst* . mask indicates which groups of state variables are to be copied. *mask* contains the bitwise OR of the same symbolic names that are passed to the OpenGL command **glPushAttrib**. The single symbolic constant **GL_ALL_ATTRIB_BITS** can be used to copy the maximum possible portion of rendering state.

Not all values for OpenGL states 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 OpenGL command **glPushAttrib**.

ERRORS

aglCopyContext returns **GL_FALSE** if it fails for any reason, **GL_TRUE** otherwise.

AGL_BAD_CONTEXT is set if either *src* or *dst* is not a valid AGL context.

OpenGL errors on either context may be generated if a renderer fails to get or set the attributes. See **glGetError**.

SEE ALSO

glPushAttrib, **aglCreateContext**

aglCreateContext

Create a new AGL rendering context.

C SPECIFICATION

```
#include <agl.h>
AGLContext aglCreateContext (
   AGLPixelFormat pix,
   AGLContext share )
```

PARAMETERS

pix	Specifies the pixel format for the new rendering context.
share	Specifies the context with which to share display lists and textures. `NULL` indicates that no sharing is to take place.

DESCRIPTION

`aglCreateContext` creates an AGL rendering context and returns its handle. This context can be used to render into a Mac OS graphics port. If *pix* was chosen with the **`AGL_OFFSCREEN`** attribute, then the context can be used to render into an off-screen graphics port.

NOTES

If *pix* was chosen to support multiple graphics devices, then the created context can render transparently across the support devices. With a multiple device context, sharing is possible only when the relationship between renderers and the graphics devices they support is the same for all contexts being shared.

ERRORS

`aglCreateContext` returns `NULL` if it fails for any reason.

`AGL_BAD_MATCH` is set if the context to be created could not share attributes with the context specified by *share*.

`AGL_BAD_CONTEXT` is set if *share* is not a valid AGL context and is not `NULL`.

`AGL_BAD_PIXELFMT` is set if *pix* is not a valid pixel format.

SEE ALSO

`aglChoosePixelFormat`, **`aglDestroyContext`**, **`aglSetDrawable`**

aglDescribePixelFormat

Return information about an AGL pixel format.

C SPECIFICATION

```
#include <agl.h>
GLboolean aglDescribePixelFormat (                    AGLPixelFormat
pix,
   GLint attrib,
   GLint *value )
```

PARAMETERS

pix	Specifies the pixel format.
attrib	Specifies the pixel format attribute to be returned.
value	Returns the requested value.

DESCRIPTION

aglDescribePixelFormat sets value to the *attrib* value of the pixel format *pix*. **aglDescribePixelFormat** returns **GL_TRUE** on successful completion.

attrib may be any of the attributes accepted by **aglChoosePixelFormat** with the except ion of **AGL_ALL_RENDERERS**, **AGL_MINIMUM_POLICY**, **AGL_MAXIMUM_POLICY**, and **AGL_CLOSEST_POLICY**, and the addition of **AGL_WINDOW**, **AGL_MULTISCREEN**, **AGL_COMPLIANT**, and **AGL_VIRTUAL_SCREEN**.

The value returned in *value* depends on the attributes, as follows:

AGL_BUFFER_SIZE
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, buffer size is the size of the color indexes.

AGL_LEVEL
Frame buffer level of the pixel format. Level zero is the default frame buffer. Positive levels correspond to frame buffers that overlay the default buffer, and negative levels correspond to frame buffers that underlay the default level.

AGL_RGBA
GL_TRUE if the color buffers store red, green, blue, and alpha values, **GL_FALSE** if they store color indexes.

AGL_DOUBLEBUFFER
GL_TRUE if color buffers exist in front/back pairs that can be swapped, **GL_FALSE** otherwise.

AGL_STEREO
GL_TRUE if color buffers exist in left/right pairs, **GL_FALSE** otherwise.

AGL_AUX_BUFFERS

The number of auxiliary buffers that are available. Zero indicates that no auxiliary buffers exist.

AGL_RED_SIZE
The number of bits of red stored in each color buffer. Zero if **AGL_RGBA** is **GL_FALSE**.

AGL_GREEN_SIZE
The number of bits of green stored in each color buffer. Zero if **AGL_RGBA** is **GL_FALSE**.

AGL_BLUE_SIZE
The number of bits of blue stored in each color buffer. Zero if **AGL_RGBA** is **GL_FALSE**.

AGL_ALPHA_SIZE
The number of bits of alpha stored in each color buffer. Zero if **AGL_RGBA** is **GL_FALSE**.

AGL_DEPTH_SIZE
Number of bits in the depth buffer

AGL_STENCIL_SIZE
The number of bits in the stencil buffer

AGL_ACCUM_RED_SIZE
The number of bits of red stored in the accumulation buffer.

AGL_ACCUM_GREEN_SIZE
The number of bits of green stored in the accumulation buffer.

AGL_ACCUM_BLUE_SIZE
The number of bits of blue stored in the accumulation buffer.

AGL_ACCUM_ALPHA_SIZE
The number of bits of alpha stored in the accumulation buffer.

AGL_PIXEL_SIZE
The number of bits of memory per pixel in the frame buffer. This value is less than or equal to the sum of red, green and blue or red, green, blue, and alpha bits because some bits in the frame buffer may not be utilized in certain modes. For off-screen rendering, the pixel size of a pixel format must be equal to the buffer depth of the off-screen rendering area.

AGL_OFFSCREEN
GL_TRUE if the pixel format can be used to render to an off-screen memory area.

AGL_FULLSCREEN
GL_TRUE if the pixel format can be used to render to a full-screen graphics device.

AGL_WINDOW
GL_TRUE if the pixel format can be used to render to a drawable window.

AGL_RENDERER_ID
The integer renderer ID of the renderer that created the pixel format.

AGL_SINGLE_RENDERER
GL_TRUE if *pix* is a single pixel format representing a single renderer, **GL_FALSE** if *pix* is a list of pixel formats representing multiple renderers.

AGL_NO_RECOVERY
GL_TRUE if failure recovery features are disabled for this pixel format.

AGL_ACCELERATED

GL_TRUE if *pix* represents a hardware accelerated renderer.

AGL_BACKING_STORE
: **GL_TRUE** if the contents of the back color buffer are guaranteed to be valid after a call to **aglSwapBuffers**, regardless of the visibility state of the current drawable.

AGL_ROBUST
: **GL_TRUE** if *pix* represents a renderer that has no failure modes associated with a lack of video resources.

AGL_MP_SAFE
: **GL_TRUE** if *pix* represents a renderer that is multi-processor safe.

AGL_COMPLIANT
: **GL_TRUE** if *pix* represents a pixel format fully compliant with OpenGL.

AGL_MULTISCREEN
: **GL_TRUE** if the pixel format can be used to render to multiple screens simultaneously. This value applies only to a particular entry in a list of pixel formats. A return of **GL_FALSE** does not imply that multiple screens are not supported, because there may be other pixel formats in the list that do provide multi-screen support.

AGL_VIRTUAL_SCREEN
: The integer virtual screen number of the pixel format. See **aglSetVirtualScreen**.

AGL_SAMPLE_BUFFERS_ARB
: The number of samples buffers of the pixel format.

AGL_SAMPLES_ARB
: The number of samples per pixel of the pixel format.

NOTES

On multi-screen systems, **aglChoosePixelFormat** may return a list of more than one pixel format to support multiple renderers simultaneously. To access the data in pixel formats after the first one in the list, use **aglNextPixelFormat**.

ERRORS

aglDescribePixelFormat returns **GL_FALSE** if it fails for any reason.

AGL_BAD_PIXELFMT is set if *pix* is not a valid pixel format.

AGL_BAD_ATTRIBUTE is set if *attrib* is not an accepted attribute.

SEE ALSO

aglChoosePixelFormat, **aglCreateContext**

aglDescribeRenderer

Return information about an AGL renderer.

C SPECIFICATION

```
#include <agl.h>
GLboolean aglDescribeRenderer ( AGLRendererInfo rend,
   GLint prop,
   GLint *value )
```

PARAMETERS

rend	Specifies the renderer info.
prop	Specifies the renderer property to be returned.
value	Returns the requested value.

DESCRIPTION

aglDescribeRenderer sets *value* to the *prop* value of the renderer info *rend*. **aglDescribeRenderer** returns **GL_TRUE** on successful completion.

prop may be any of the following symbolic values:

AGL_RENDERER_ID
The integer renderer ID of the renderer that created the pixel format.

AGL_OFFSCREEN
GL_TRUE if the renderer can render to an off-screen memory area.

AGL_FULLSCREEN
GL_TRUE if the renderer can render to a full-screen graphics device.

AGL_WINDOW
GL_TRUE if the renderer can render to a drawable window.

AGL_ACCELERATED
GL_TRUE if the renderer is hardware accelerated.

AGL_BACKING_STORE
GL_TRUE if the contents of a back color buffer are guaranteed to be valid after a call to **aglSwapBuffers**, regardless of the visibility state of the current drawable.

AGL_ROBUST
GL_TRUE if the renderer has no failure modes associated with a lack of video resources.

AGL_MP_SAFE
GL_TRUE if the renderer is multi-processor safe.

AGL_COMPLIANT
GL_TRUE if the renderer is fully compliant with the OpenGL specification.

AGL_MULTISCREEN

GL_TRUE if the renderer is capable of driving multiple screens with the same rendering context. This value may affect the way **aglChoosePixelFormat** chooses renderers to support multiple screens.

AGL_BUFFER_MODES

The bitwise OR of the following frame buffer mode flags:

AGL_MONOSCOPIC_BIT

AGL_STEREOSCOPIC_BIT

AGL_SINGLEBUFFER_MODE

AGL_DOUBLEBUFFER_MODE

AGL_MIN_LEVEL

The minimum overlay buffer level. Negative values indicate an underlay buffer.

AGL_MAX_LEVEL

The maximum overlay buffer level.

AGL_COLOR_MODES

AGL_ACCUM_MODES

Either of these properties can be the bitwise OR of any of the following symbolic values:

AGL_RGB8_BIT

AGL_RGB8_A8_BIT

AGL_BGR233_BIT

AGL_BGR233_A8_BIT

AGL_RGB332_BIT

AGL_RGB332_A8_BIT

AGL_RGB444_BIT

AGL_ARGB4444_BIT

AGL_RGB444_A8_BIT

AGL_RGB555_BIT

AGL_ARGB1555_BIT

AGL_RGB555_A8_BIT

AGL_RGB565_BIT

AGL_RGB565_A8_BIT

AGL_RGB888_BIT

AGL_ARGB8888_BIT

AGL_RGB888_A8_BIT

AGL_RGB101010_BIT

AGL_ARGB2101010_BIT

AGL_RGB101010_A8_BIT

AGL_RGB121212_BIT

AGL_ARGB12121212_BIT

AGL_RGB161616_BIT

AGL_ARGB16161616_BIT

AGL_INDEX8_BIT

AGL_INDEX16_BIT

AGL_DEPTH_MODES

AGL_STENCIL_MODES
Any of these properties can be the bitwise `OR` of any of the following flags:

AGL_0_BIT

AGL_1_BIT

AGL_2_BIT

AGL_4_BIT

AGL_8_BIT

AGL_12_BIT

AGL_16_BIT

AGL_24_BIT

AGL_32_BIT

AGL_48_BIT

AGL_64_BIT

AGL_MAX_AUX_BUFFERS
The maximum number of auxiliary buffers that can be supported by the renderer.

AGL_VIDEO_MEMORY
The amount of physical video memory (in bytes) associated with the renderer. This will be reported as zero for software based renderers.

AGL_TEXTURE_MEMORY
The amount of physical memory (in bytes) specifically allocated for texture storage associated with the renderer. If the renderer has a unified memory model and does not support specific texture memory, this will be reported the same as **AGL_VIDEO_MEMORY.** This will be reported as zero for software based renderers.

NOTES

aglQueryRendererInfo will normally return a list of more than one renderer info; one for each renderer found on the system. To access the data in renderer infos after the first one in the list, use **aglNextRendererInfo**.

ERRORS

aglDescribeRenderer returns **GL_FALSE** if it fails for any reason.

AGL_BAD_RENDINFO is set if *rend* is not a valid renderer info.

`AGL_BAD_PROPERTY` is set if *prop* is not an accepted property.

SEE ALSO

`aglQueryRendererInfo`, **`aglNextRendererInfo`**

aglDestroyContext

Destroy an AGL rendering context.

C SPECIFIC ATION

```
#include <agl.h>
GLboolean aglDestroyContext ( AGLContext ctx )
```

PARAMETERS

ctx	Specifies the AGL context to be destroyed.

DESCRIPTION

If the AGL rendering context *ctx* is the current rendering context, then there will be no current context after **`aglDestroyContext`** executes. All resources used by *ctx* are freed immediately. **`aglDestroyContext`** returns **`GL_TRUE`** on successful completion.

ERRORS

`aglDestroyContext` returns **`GL_FALSE`** if it fails for any reason

`AGL_BAD_CONTEXT` is set if *ctx* is not a valid AGL context.

SEE ALSO

`aglCreateContext`, **`aglUpdateContext`**

aglDestroyPixelFormat

Free resources used by a pixel format.

C SPECIFICATION

```
#include <agl.h>
void aglDestroyPixelFormat ( AGLPixelFormat pix )
```

PARAMETERS

pix	Specifies the pixel format to be destroyed.

DESCRIPTION

`aglDestroyPixelFormat` frees the memory allocated by **`aglChoosePixelFormat`**. A copy of the pixel format data is made by **`aglCreateContext`**, so an application may free a pixel format immediately after creating a context with it.

Do not pass the return from **`aglNextPixelFormat`** to **`aglDestroyPixelFormat`**. Doing so will set the **`AGL_BAD_PIXELFMT`** error.

ERRORS

`AGL_BAD_PIXELFMT` is set if *pix* is not a valid pixel format.

SEE ALSO

`aglChoosePixelFormat`, **`aglDescribePixelFormat`**

aglDestroyRendererInfo

Free resources used by a renderer info.

C SPECIFICATION

```
#include <agl.h>
void aglDestroyRendererInfo ( AGLRendererInfo rend )
```

PARAMETERS

rend	Specifies the renderer info to be destroyed.

DESCRIPTION

`aglDestroyRendererInfo` frees the memory allocated by **`aglQueryRendererInfo`**. Specific information is obtained from a renderer info with **`aglDescribeRendererInfo`**.

Do not pass the return from **`aglNextRendererInfo`** to **`aglDestroyRendererInfo`**. Doing so will set the **`AGL_BAD_RENDINFO`** error.

ERRORS

`AGL_BAD_RENDINFO` is set if *rend* is not a valid renderer info.

SEE ALSO

`aglQueryRendererInfo`, **`aglDescribeRendererInfo`**

aglDevicesOfPixelFormat

Return the graphics devices supported by a pixel format.

C SPECIFICATION

```
#include <agl.h>
AGLDevice *aglDevicesOfPixelFormat ( AGLPixelFormat pix,
    GLint *ndevs )
```

PARAMETERS

pix	Specifies the pixel format.
ndevs	Returns the number of devices in the returned array.

RETURN

An array of graphics device specifiers of length *ndevs*.

DESCRIPTION

`aglChoosePixelFormat` may return a list of more than one pixel format. The first format in the list is guaranteed to support all of the graphics devices requested of **`aglChoosePixelFormat`**. However, all subsequent devices in the list will support only a non-overlapping subset of all requested graphics devices. The devices supported by each pixel format can be determined with **`aglNextPixelFormat`** and **`aglDevicesOfPixelFormat`**.

The AGL library manages switching between the renderers that support each graphics devices. An application should only be concerned with the information provided by this function if it wishes to implement alternative rendering modes for specific renderers.

ERRORS

`aglDevicesOfPixelFormat` returns `NULL` if it fails for any reason

`AGL_BAD_PIXELFMT` is set if *pix* is not a valid pixel format.

SEE ALSO

`aglChoosePixelFormat`, **`aglDescribePixelFormat`**, **`aglNextPixelFormat`**

aglDisable

Disable an AGL context option.

C SPECIFICATION

```
#include <agl.h>
GLboolean aglDisable ( AGLContext ctx ,
   GLenum pname )
```

PARAMETERS

ctx	Specifies the AGL context.
pname	Specifies the capability to be disabled.

DESCRIPTION

aglDisable disables an AGL option that was enabled with **aglEnable**. *pname* may be any one of the symbolic constants accepted by **aglEnable**. **aglDisable** returns **GL_FALSE** if it fails for any reason, **GL_TRUE** otherwise.

ERRORS

AGL_BAD_CONTEXT is set if *ctx* is not a valid context.

AGL_BAD_ENUM is set if *pname* is not one of the accepted values.

SEE ALSO

glEnable, **aglEnable**

aglEnable

Enable an AGL context option.

C SPECIFICATION

```
#include <agl.h>
GLboolean aglEnable (  AGLContext ctx ,
   GLenum pname )
```

PARAMETERS

ctx	Specifies the AGL context.
pname	Specifies the capability to be enabled.

DESCRIPTION

`aglEnable` enables an AGL option. Use **`aglDisable`** to disable the option. **`aglEnable`** returns **`GL_FALSE`** if it fails for any reason, **`GL_TRUE`** otherwise.

pname may be one of the following symbolic constants:

`AGL_SWAP_RECT`
If enabled, the area of the window that is affected by **`aglSwapBuffers`** is restricted to a sub-rectangle of the entire window.

`AGL_BUFFER_RECT`
If enabled, the drawable rectangle of the window and all of its associated buffers are restricted to a rectangle specified with **`aglSetInteger`**.

`AGL_CLIP_REGION`
If enabled, the drawable of the window and all of its associated buffers are restricted to the QuickDraw region specified with **`aglSetInteger`**. **(New in AGL for Max OS X 10.2 Jaguar)**

`AGL_RASTERIZATION`
If disabled, all rasterization of 2D and 3D primitives will be disabled. This state is useful for debugging and to characterize the performance of an OpenGL driver without actually rendering.

`AGL_STATE_VALIDATION`
If enabled, the AGL library will inspect the context state each time that **`aglUpdateContext`** is called to ensure that it is in an appropriate state for switching between renderers. Normally, the state is inspected only when it is actually necessary to switch renderers. This is useful to use a single monitor system to test that an application will perform correctly on a multiple monitor system.

`AGL_COLORMAP_TRACKING`
If enabled, a rendering context of 8-bit depth (RGBA or color index format) uses the color table associated with the MacOS window to which it is attached. For RGBA formats, a change to the window's color table must be followed by a call to **`aglUpdateContext`** to inform the context that the

color table has changed. For color index formats, the window's color table may be changed at any time. This mode offers the best performance since color translation does not occur when data is copied from the color buffer to the window. If disabled, the rendering context uses an internal color table that is defined by calling **`aglSetInteger`** with the **`AGL_COLORMAP_ENTRY`** parameter name.

`AGL_SWAP_LIMIT`

If enabled, calls the aglSwapBuffers will block when the previous swap has not completed, thus providing a synchronization primitive which does not let the application get more than single frame ahead of the graphics hardware. If disabled, this synchronization is removed this makes possible for an application to get significantly ahead of graphics hardware. The normal case is for this to be enabled. **(New in AGL for Max OS X 10.2 Jaguar)**

`AGL_FS_CAPTURE_SINGLE`

If enabled, **`aglFullScreen`** will only capture a single display. If disabled, all displayed will be captured, thus preventing other applications from being affected by resolution switches. The normal case is for this to be disabled. **(New in AGL for Max OS X 10.2 Jaguar)**

ERRORS

`AGL_BAD_CONTEXT` is set if *ctx* is not a valid context.

`AGL_BAD_ENUM` is set if *pname* is not one of the accepted values.

SEE ALSO

`glEnable`, **`aglDisable`**, **`aglIsEnabled`**, **`aglSetInteger`**

aglErrorString

Return an error string for an AGL error code.

C SPECIFICATION

```
#include <agl.h>
const GLubyte *aglErrorString ( GLenum code )
```

PARAMETERS

code	Specifies an AGL error code.

DESCRIPTION

aglErrorString produces an error string from an AGL error code. The standard AGL error codes are **AGL_NO_ERROR** and all the numerical codes between **AGL_BAD_ATTRIBUTE** and **AGL_BAD_ALLOC**, inclusive.

aglErrorString always returns a string, even if *code* is invalid.

SEE ALSO

aglGetError, **gluErrorString**

aglGetCurrentContext

Return the current context.

C SPECIFICATION

```
#include <agl.h>
AGLContext aglGetCurrentContext ( void )
```

DESCRIPTION

`aglGetCurrentContext` returns the current AGL rendering context, as specified by **`aglSetCurrentContext`**. If there is no current context, `NULL` is returned.

SEE ALSO

`aglCreateContext`, **`aglSetCurrentContext`**

aglGetDrawable

Return the drawable attached to a rendering context.

C SPECIFICATION

```
#include <agl.h>
AGLDrawable aglGetDrawable ( AGLContext ctx )
```

PARAMETERS

ctx	Specifies the rendering context.

DESCRIPTION

aglGetDrawable returns the AGL drawable (a Mac OS `CGrafPtr`) that was last attached to *ctx* with **aglSetDrawable**.

If the drawable last attached to *ctx* was an off-screen drawable (attached with **aglSetOffScreen**) **aglGetDrawable** returns the base address of the off-screen memory area. If the drawable last attached to *ctx* was a full-screen graphics device (attached with **aglSetFullScreen**) **aglGetDrawable** returns the integer device number of the full-screen graphics device.

aglGetDrawable returns `NULL` if no drawable is attached to *ctx*.

ERRORS

aglGetDrawable returns `NULL` if it fails for any reason.

AGL_BAD_CONTEXT is set if *ctx* is not a valid context.

SEE ALSO

aglCreateContext, **aglSetDrawable**, **aglSetFullScreen**, **aglSetOffScreen**

aglGetError

Return error information.

C SPECIFICATION

```
#include <agl.h>
GLenum aglGetError ( void )
```

DESCRIPTION

aglGetError returns the value of the global AGL error flag. Each error is assigned a numeric code and 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**, there has been no detectable error since the last call to **aglGetError**.

The currently defined errors are as follows:

AGL_NO_ERROR	No error.
AGL_BAD_ATTRIBUTE	Unknown pixel format attribute.
AGL_BAD_PROPERTY	Unknown renderer property.
AGL_BAD_PIXELFMT	Invalid pixel format specified.
AGL_BAD_RENDINFO	Invalid renderer info.
AGL_BAD_CONTEXT	Invalid context specified.
AGL_BAD_DRAWABLE	Invalid drawable specified.
AGL_BAD_GDEV	Invalid graphics device.
AGL_BAD_STATE	Operation not allowed in current state.
AGL_BAD_VALUE	Out of range numerical value.
AGL_BAD_MATCH	Contexts cannot be shared.
AGL_BAD_ENUM	Invalid enumeration
AGL_BAD_OFFSCREEN	Invalid off-screen drawable specification

`AGL_BAD_FULLSCREEN`
: Invalid full-screen drawable specification

`AGL_BAD_WINDOW`
: Invalid drawable window specification

`AGL_BAD_POINTER`
: Null pointer encountered

`AGL_BAD_MODULE`
: Invalid code module loaded.

`AGL_BAD_ALLOC`
: Memory allocation failure has occurred.

SEE ALSO

`glGetError`

aglGetInteger

Retrieve the integer settings of an AGL context option.

C SPECIFICATION

```
#include <agl.h>
GLboolean aglGetInteger ( AGLContext ctx,
   GLenum pname,
   GLint *params )
```

PARAMETERS

ctx	Specifies the AGL context.
pname	Specifies the option settings to be returned.
params	Returns the option settings.

DESCRIPTION

aglGetInteger returns the current setting of an AGL option. Use **aglSetInteger** to alter the setting and **aglEnable** to enable the option. **aglGetInteger** returns **GL_FALSE** if it fails for any reason, **GL_TRUE** otherwise.

pname may be one of the following symbolic constants:

AGL_SWAP_RECT
params returns four values: the x and y window coordinates of the swap rectangle, followed by its width and height.

AGL_BUFFER_RECT
params returns four values: the x and y window coordinates of the buffer rectangle, followed by its width and height.

AGL_CLIP_REGION
On entry, *params* should contain a pointer to a valid region handle. On return, the region handle will contain a copy of the clipping region used to clip the drawable. Applications must ensure the region handle is initialized before calling **aglGetInteger. (New in AGL for Max OS X 10.2 Jaguar)**

AGL_OFFSCREEN
If the drawable currently attached to *ctx* is an off-screen drawable (attached with **aglSetOffScreen**) *params* returns three values: the width, height, and rowbytes of the off-screen memory area. If the drawable of *ctx* is not an off-screen type, *params* returns zeroes.

AGL_FULLSCREEN
If the drawable currently attached to *ctx* is a full-screen drawable (attached with **aglSetFullScreen**), *params* returns three values: the width, height, and refresh frequency of the full-screen device. If the drawable of *ctx* is not a full-screen type, *params* returns zeroes.

AGL_SWAP_INTERVAL

params returns one value: the current swap interval setting.

`AGL_COLORMAP_ENTRY`
params[0] must be initialized to a valid color index on entry. On return, *params*[1], *params*[2], and *params*[3] contain the red, green, and blue intensities of the specified color table entry. The return values are scaled so minimum intensity maps to 0 and maximum intensity maps to 65535.

`AGL_BUFFER_NAME`
params returns one value: the name (a positive integer) of the buffer associated with the current context. This provides a mechanism to map multiple buffers to a single context. See **`aglSetInteger`** for more details.

`AGL_CONTEXT_SURFACE_ID`
This constant has been deprecated.

`AGL_CONTEXT_DISPLAY_ID`
This constant has been deprecated.

`AGL_SURFACE_ORDER`
params returns one value: the current position of the surface relative to the associated window, with 1 be the normal position above the window and –1 meaning the surface is below the window. **(New in AGL for Max OS X 10.2 Jaguar)**

`AGL_SURFACE_OPACITY`
params returns one value: the current surface opacity, with 1 being opaque and 0 being transparent. If this value is 0, the opacity is determined on a per pixel basis depending on the frame buffer's pixel alpha value. **(New in AGL for Max OS X 10.2 Jaguar)**

ERRORS

`AGL_BAD_CONTEXT` is set if *ctx* is not a valid context.

`AGL_BAD_ENUM` is set if *pname* is not one of the accepted values.

SEE ALSO

`aglEnable`, **`aglSetInteger`**

aglGetVersion

Return the version numbers of the AGL library.

C SPECIFICATION

```
#include <agl.h>
GLboolean aglGetVersion ( GLint *major,
   GLint *minor )
```

PARAMETERS

major	Returns the major version number of the AGL library.
minor	Returns the minor version number of the AGL library.

DESCRIPTION

`aglGetVersion` returns the major and minor version numbers of the AGL library. AGL implementations with the same major version number are upward compatible, meaning that the implementation with the higher minor number is a superset of the version with the lower minor number. This is not the version of OpenGL renderer associated with a specific context and drawable. The OpenGL version of the current renderer can be found using the **`glGetString`** function once a context and drawable have been established.

major and *minor* do not return values if they are specified as `NULL`.

SEE ALSO

`glGetString`

aglGetVirtualScreen

Return the current virtual screen number.

C SPECIFICATION

```
#include <agl.h>
GLint aglGetVirtualScreen ( AGLContext ctx )
```

PARAMETERS

ctx	Specifies the AGL context.

DESCRIPTION

aglGetVirtualScreen may be used on multiple-monitor systems to find which virtual screen is associated with the OpenGL renderer that is currently processing OpenGL commands. On a single-monitor system, **aglGetVirtualScreen** always returns zero. The current virtual screen is normally set automatically by **aglUpdateCurrent** to be the virtual screen that contains the greatest area of the drawable, so the current virtual screen may change when the drawable is moved or resized across graphics device boundaries. A change in the current virtual screen may affect the return values and/or validity of some OpenGL functions (especially extensions).

NOTES

Each virtual screen is equivalent to a Mac OS X graphics devices (not display devices, but graphics hardware devices). The total number of virtual screens is less than or equal to the number of graphics devices. There is one OpenGL renderer and one pixel format associated with each virtual screen. Note, OpenGL commands are always processed by the renderer associated with the current virtual screen. The relationship between virtual screens and their respective renderers and pixel formats is determined entirely by **aglChoosePixelFormat**.

The virtual screen number and OpenGL renderer ID associated with a specific pixel format are found by passing **aglDescribePixelFormat** the **AGL_VIRTUAL_SCREEN** and **AGL_RENDERER_ID** attributes, respectively, and the set of graphics devices associated with a pixel format is found with **aglDevicesOfPixelFormat**. **aglNextPixelFormat** and **aglDescribePixelFormat** can be used repeatedly to examine all the pixel formats returned by **aglChoosePixelFormat**.

ERRORS

aglGetVirtualScreen returns -1 if it fails for any reason.

AGL_BAD_CONTEXT is set if *ctx* is not a valid context.

SEE ALSO

aglChoosePixelFormat, **aglDescribePixelFormat**,

aglDevicesOfPixelFormat, **aglNextPixelFormat**, **aglSetVirtualScreen**

aglIsEnabled

Query the state of an AGL context option.

C SPECIFICATION

```
#include <agl.h>
GLboolean aglIsEnabled (  AGLContext ctx ,
   GLenum pname )
```

PARAMETERS

ctx	Specifies the AGL context.
pname	Specifies the capability to be queried.

DESCRIPTION

aglIsEnabled queries the state of an AGL option that was enabled or disabled with **aglEnable** or **aglDisable**. *pname* may be any one of the symbolic constants accepted by **aglEnable**. **aglIsEnabled** returns **GL_TRUE** if the option is enabled, **GL_FALSE** if the option is disabled or if an error occurs.

ERRORS

AGL_BAD_CONTEXT is set if *ctx* is not a valid context.

AGL_BAD_ENUM is set if *pname* is not one of the accepted values.

SEE ALSO

aglDisable, **aglEnable**, **aglGetInteger**, **aglSetInteger**

aglNextPixelFormat

Return the next in a list of pixel formats.

C SPECIFICATION

```
#include <agl.h>
AGLPixelFormat aglNextPixelFormat ( AGLPixelFormat *pix )
```

PARAMETERS

pix	Specifies a pixel format.

DESCRIPTION

`aglNextPixelFormat` returns the next pixel format in a list of pixel formats. If *pix* is the last pixel format in the list, `NULL` is returned.

NOTES

Lists of more than one pixel format are generated by **`aglChoosePixelFormat`** when not all the graphics devices on the system are supported by a single renderer.

ERRORS

`aglNextPixelFormat` returns `NULL` if it fails for any reason.

`AGL_BAD_PIXELFMT` is set if *pix* is not a valid AGL pixel format.

SEE ALSO

`aglChoosePixelFormat`, **`aglDescribePixelFormat`**

aglNextRendererInfo

Return the next in a list of renderer infos.

C SPECIFICATION

```
#include <agl.h>
AGLRendererInfo aglNextRendererInfo (            AGLRendererInfo
*rend )
```

PARAMETERS

rend	Specifies a renderer info.

DESCRIPTION

`aglNextRendererInfo` returns the next renderer info in a list of renderer infos. If *rend* is the last renderer info in the list, `NULL` is returned.

NOTES

Lists of more than one renderer info are generated by **`aglQueryRendererInfo`** when there is more than one renderer installed on the system. Most systems have more than one installed renderer since support for different buffer depths is often provided by separate renderers. This function simply iterates the list allocated by **`aglDescribeRenderer`** and does not allocate any memory.

ERRORS

`aglNextRendererInfo` returns `NULL` if it fails for any reason.

`AGL_BAD_RENDINFO` is set if *rend* .is not a valid AGL renderer info.

SEE ALSO

`aglQueryRendererInfo`, **`aglDescribeRenderer`**

aglQueryRendererInfo

Retrieve a description of renderer capabilities.

C SPECIFICATION

```
#include <agl.h>
AGLRendererInfo aglQueryRendererInfo (                    const
AGLDevice *gdev,
   GLint ndev )
```

PARAMETERS

gdev	An array of Mac OS graphics devices (type `GDHandle`)
ndev	The number of graphics devices in *gdev*

DESCRIPTION

`aglQueryRendererInfo` returns a list of AGLRendererInfo data structures that describe the capabilities of OpenGL renderers. One AGLRendererInfo is returned for each OpenGL rendering engine installed on the system. To access the AGLRendererInfo data, use **`aglDescribeRenderer`**. To free the data returned by this function, use **`aglDestroyRendererInfo`**.

If *gdev* and *ndev* are `NULL` and zero, respectively, the returned information will apply to all graphics devices on the system. Otherwise, information will be returned for only the specified devices.

ERRORS

`aglQueryRendererInfo` returns `NULL` if it fails for any reason.

`AGL_BAD_DEVICE` is set if *ndev* is nonzero and *gdev* is not an array of valid devices.

SEE ALSO

`aglChoosePixelFormat`, **`aglDescribeRenderer`**, **`aglDestroyRendererInfo`**, **`aglNextRendererInfo`**

aglResetLibrary

Reset the OpenGL library to its initial state.

C SPECIFICATION

```
#include <agl.h>
void aglResetLibrary ( void )
```

DESCRIPTION

aglResetLibrary resets the OpenGL library to its initial state. **aglResetLibrary** destroys all contexts created with **aglCreateContext**, unloads all plug-in renderers from memory, frees any data allocated by **aglChoosePixelFormat** or **aglQueryRendererInfo**, and resets any options set with **aglConfigure** to their initial values.

If any resources have been allocated by the OpenGL library, **aglResetLibrary** must be called to free those resources before attempting to change the memory page allocation mode of the OpenGLMemory library.

SEE ALSO

aglConfigure, **aglDestroyContext**, **aglDestroyPixelFormat**, **aglDestroyRendererInfo**

aglSetCurrentContext

Make a context the current rendering context.

C SPECIFICATION

```
#include <agl.h>
GLboolean aglSetCurrentContext (                          AGLContext
ctx )
```

PARAMETERS

ctx	Specifies an AGL rendering context.

DESCRIPTION

aglSetCurrentContext makes *ctx* the current AGL rendering context, replacing the previously current context if there was one. Because of this action, subsequent OpenGL rendering calls go to rendering context *ctx* to modify its drawable. Because **aglSetCurrentContext** always replaces the current rendering context with *ctx* , there can be only one current context.

To release the current context without assigning a new one, call **aglSetCurrentContext** with *ctx* set to `NULL`.

If **aglSetCurrentContext** fails, the current rendering context remains unchanged.

ERRORS

aglSetCurrentContext returns `GL_FALSE` if it fails for any reason.

AGL_BAD_CONTEXT is set if *ctx* is not a valid AGL context and is not `NULL`.

SEE ALSO

aglCreateContext, **aglGetCurrentContext**, **aglSetDrawable**

aglSetDrawable

Attach an AGL context to a Mac OS graphics port.

C SPECIFICATION

```
#include <agl.h>
GLboolean aglSetDrawable (   AGLContext ctx,
   AGLDrawable draw )
```

PARAMETERS

ctx	Specifies an AGL rendering context.
draw	Specifies an AGL drawable. The **`AGLDrawable`** type is equivalent to the Mac OS **`CGrafPtr`** type.

DESCRIPTION

`aglSetDrawable` attaches drawable *draw* to rendering context *ctx*. Because of this action, subsequent OpenGL rendering calls directed to *ctx* modify drawable *draw*. **`aglSetDrawable`** performs all of the actions performed by **`aglUpdateContext`**.

When a context is first attached to a specific drawable, its viewport is set to the full size of the drawable. If the context is subsequently attached to the same drawable, its viewport is unaltered.

To disable a rendering context, call **`aglSetDrawable`** with draw set to `NULL`.

If **`aglSetDrawable`** fails, the drawable of the context is set to `NULL`.

ERRORS

`aglSetDrawable` returns **`GL_FALSE`** if it fails for any reason.

`AGL_BAD_DRAWABLE` is set if *draw* is not a valid AGL drawable or `NULL`.

`AGL_BAD_CONTEXT` is set if *ctx* is not a valid AGL context.

SEE ALSO

`aglCreateContext`, **`aglSetCurrentContext`**

aglSetFullScreen

Attach an AGL context to a full-screen graphics device.

C SPECIFICATION

```
#include <agl.h>
GLboolean aglSetFullScreen ( AGLContext ctx,
    GLsizei width,
    GLsizei height,
    GLsizei freq,
    GLint device )
```

PARAMETERS

ctx	Specifies an AGL rendering context.
width	Specifies the width of the graphics device in pixels.
height	Specifies the height of the graphics device in pixels.
freq	Specifies the refresh frequency of the graphics device in hertz.
device	Specifies the integer graphics device index.

DESCRIPTION

aglSetFullScreen attaches context *ctx* to a full-screen graphics device, which was previously specified at pixel format creation. Because of this action, subsequent OpenGL rendering calls directed to *ctx* modify the full screen context. The context must have been created with respect to a pixel format that supports a full-screen device, which is requested with the **AGL_FULLSCREEN** attribute for **aglChoosePixelFormat**. **aglSetFullScreen** performs all of the actions performed by **aglUpdateContext**.

Developers should not assume the screen resulting resolution is exactly as requested, due to such factors as video mirroring. It is strongly recommended applications test the current resolution upon return ensuring they are using the current resolution. The resolution set can be found by using **aglGetInteger** with the **AGL_FULLSCREEN** parameter or other Mac OS X routines, which return the current display resolution.

When a context is first attached to a full-screen device, its viewport is set to the current resolution of the device. If the context is subsequently attached to the same device, its viewport is unaltered. In addition, full screen contexts may use page flipping to update their content vice buffer copies, thus affecting the existence of a valid backing store.

The integer device number is currently ignored on Mac OS X.

To disable a rendering context, call **aglSetDrawable** with *draw* set to `NULL`.

If **aglSetFullScreen** fails, the drawable of the context is set to `NULL`.

ERRORS

aglSetFullScreen returns **GL_FALSE** if it fails for any reason.

AGL_BAD_FULLSCREEN is set if *width*, *height*, or *freq* are not supported by the device.

AGL_BAD_CONTEXT is set if *ctx* is not a valid AGL context.

SEE ALSO

aglCreateContext, **aglSetCurrentContext**, **aglSetDrawable**

aglSetInteger

Set the integer values of AGL context options.

C SPECIFICATION

```
#include <agl.h>
GLboolean aglSetInteger ( AGLContext ctx,
   GLenum pname,
   GLint *params )
```

PARAMETERS

ctx	Specifies the AGL context.
pname	Specifies that option values are to be changed.
params	A pointer to the new option values.

DESCRIPTION

aglSetInteger changes the current setting of an AGL context option. Use **aglGetInteger** to retrieve the setting and **aglEnable** to enable the option. **aglSetInteger** returns **GL_FALSE** if it fails for any reason, **GL_TRUE** otherwise.

pname may be one of the following symbolic constants:

AGL_SWAP_RECT

params contains four values: the x and y window coordinates of the swap rectangle, followed by its width and height. When **AGL_SWAP_RECT** is enabled, the actual screen area swapped by **aglSwapBuffers** will be restricted to the intersection of the specified rectangle and the drawable rectangle. The swap rectangle is defined in OpenGL screen coordinates, not operating system screen coordinates.

If the buffer rectangle is also enabled, the swap rectangle coordinates are relative to the buffer rectangle, not the window.

AGL_BUFFER_RECT

params contains four values: the x and y window coordinates (bottom left origin) of the buffer rectangle, followed by its width and height. The specified buffer rectangle is clamped the maximum drawable width and height and the resulting rectangle is the drawable rectangle for all GL operations. All internally allocated buffers are allocated to match the buffer rectangle, not the actual window rectangle.

If all OpenGL drawing is to be restricted to a sub-rectangle of the entire window, it is more efficient and simpler to use **AGL_BUFFER_RECT** than to use a combination of **glViewport**, **glScissor**, and **AGL_SWAP_RECT**. The buffer rectangle can be used to emulate child windows provided by some windowing systems.

AGL_CLIP_REGION

params contains a single value: a pointer to a valid region handle. The specified region is used to limit the surface shape of the current context. All internally allocated buffers are allocated to still match the window rectangle with the addition of the region providing a surface shape. Since this shape will not be drawn, whatever content is present in the window in the areas not covered by the region will be drawn and updated. It should be noted, complicated regions may affect drawing performance. **(New in AGL for Max OS X 10.2 Jaguar)**

`AGL_SWAP_INTERVAL`

params contains one value, the current swap interval setting. If the swap interval is set to 0 (the default) a call to **`aglSwapBuffers`** will be executed as soon as possible, without regard to the vertical refresh rate of the monitor. If the swap interval is set to 1 or greater, the buffers will be swapped only during the vertical retrace of the monitor. Effectively this currently acts as a switch for VBL synchronizing. Calls to **`aglSwapBuffers`** that occur at a higher rate than the monitor does refresh block until the next vertical blank after completion of the previous buffer swap (assuming **`AGL_SWAP_LIMIT`** is set to it's default). If **`AGL_SWAP_LIMIT`** is set to **`GL_FALSE`**, **`aglSwapBuffers`** will normally return immediately though the actual swap will wait for the next vertical blank after the previously queued swap completes.

`AGL_COLORMAP_ENTRY`

params contains four values: a color table index and the red, green, and blue color intensities to assign to the specified color table index. The color intensity values are scaled so 0 maps to minimum intensity and 65535 maps to maximum intensity. The color table entries set with **`AGL_COLORMAP_ENTRY`** have no effect unless **`AGL_COLORMAP_TRACKING`** is disabled.

`AGL_ORDER_CONTEXT_TO_FRONT`

params is ignored. Will order the surface associated with the context to the front of all other context attached to the current drawable. **(New in AGL for Max OS X 10.2 Jaguar)**

`AGL_BUFFER_NAME`

params contains one value: a non-negative integer name of the surface to be associated to be with the current context. If this value is non-zero, and a surface of this name is not associated to this drawable, a new surface with this name is created and associated with the context when **`aglSetDrawable`** is called subsequently. If this is a previously allocated buffer name within the namespace of the current window (e.g., drawable), that previously allocated surface is associated with the context (e.g., no new surface is created) and the subsequent call to **`aglSetDrawable`** will attach that surface. This allows multiple contexts to be attached to a single surface. Using the default buffer name zero, returns to one surface per context behavior. **(New in AGL for Max OS X 10.2 Jaguar)**

`AGL_SURFACE_ORDER`

params contains one value: the position of the surface relative to the window content. If this is 1 the surface is this normal position above the window. If this is –1 the surface is placed under the window and transparent areas of the window will show through to the surface. **(New in AGL for Max OS X 10.2 Jaguar)**

`AGL_SURFACE_OPACITY`

params contains one value: the surface opacity settings. If this value is 1 the surface is considered opaque. If the opacity value is 0 the surface will use the actual pixel alpha value. **(New in AGL for Max OS X 10.2 Jaguar)**

ERRORS

aglSetInteger returns **GL_FALSE** if it fails for any reason.

AGL_BAD_CONTEXT is set if *ctx* is not a valid context.

AGL_BAD_ENUM is set if *pname* is not one of the accepted values.

SEE ALSO

aglEnable, **aglGetInteger**, **aglSwapBuffers**

aglSetOffScreen

Attach an AGL context to an off-screen memory area.

C SPECIFICATION

```
#include <agl.h>
GLboolean aglSetOffScreen (  AGLContext ctx,
   GLsizei width,
   GLsizei height,
   GLsizei rowbytes,
   GLvoid *baseaddr )
```

PARAMETERS

ctx	Specifies an AGL rendering context.
width	Specifies the width of the off-screen memory area in pixels.
height	Specifies the height of the off-screen memory area in pixels.
rowbytes	Specifies the number of bytes in one row of the off-screen memory area.
baseaddr	Specifies the base address of the memory area.

DESCRIPTION

aglSetOffScreen attaches context *ctx* to an off-screen memory area. Because of this action, subsequent OpenGL rendering calls directed to *ctx* modify the off-screen memory. The context must have been created with respect to a pixel format that supports off-screen rendering, which is requested with the **AGL_OFFSCREEN** attribute for **aglChoosePixelFormat**. **aglSetOffScreen** also performs all of the actions performed by **aglUpdateContext**.

When a context is attached to an off-screen memory area, its viewport is set to the full size of the off-screen area.

To disable a rendering context, call **aglSetDrawable** with draw set to `NULL`.

If **aglSetOffScreen** fails, the drawable of the context is set to `NULL`.

NOTES

Currently only the software renderer supports the **AGL_OFFSCREEN** pixel attribute.

ERRORS

aglSetOffScreen returns **GL_FALSE** if it fails for any reason.

AGL_BAD_OFFSCREEN is set if the combination of *width* and *rowbytes* do not support the pixel size of the context.

AGL_BAD_CONTEXT is set if *ctx* is not a valid AGL context.

`AGL_BAD_DRAWABLE` is set if *rowbytes* is insufficient to store a row of pixels.

SEE ALSO

`aglCreateContext`, **`aglSetCurrentContext`**, **`aglSetDrawable`**

aglSetVirtualScreen

Force subsequent OpenGL commands to go to the specified virtual screen

C SPECIFICATION

```
#include <agl.h>
GLboolean aglSetVirtualScreen ( AGLContext ctx,
   GLint screen )
```

PARAMETERS

ctx	Specifies the AGL context.
screen	Specifies the virtual screen number.

DESCRIPTION

`aglSetVirtualScreen` may be used on multiple-monitor systems to specify the virtual screen and associated OpenGL renderer that will subsequently process OpenGL commands. The current virtual screen is normally set automatically by **`aglSetDrawable`** or **`aglUpdateContext`** to be the virtual screen that includes the smallest set of graphics devices that contain the entire drawable. **`aglSetVirtualScreen`** should be used only when it is necessary to override the default behavior.

NOTES

Each virtual screen includes one or more Mac OS graphics devices. Virtual screen zero of a particular AGL context always includes all graphics devices that are supported by the context and all other virtual screens include non-intersecting subsets of those devices. The total number of virtual screens is less than or equal to the number of graphics devices plus one. There is one OpenGL renderer and one pixel format associated with each virtual screen - OpenGL commands are processed by the renderer associated with the current virtual screen. The relationship between virtual screens and their respective renderers and pixel formats is determined entirely by **`aglChoosePixelFormat`**.

The virtual screen number and OpenGL renderer ID associated with a specific pixel format are found by passing **`aglDescribePixelFormat`** the **`AGL_VIRTUAL_SCREEN`** and **`AGL_RENDERER_ID`** attributes, respectively, and the set of graphics devices associated with a pixel format is found with **`aglDevicesOfPixelFormat`**. **`aglNextPixelFormat`** and **`aglDescribePixelFormat`** can be used repeatedly to examine all the pixel formats returned by **`aglChoosePixelFormat`**.

Because the current virtual screen determines which OpenGL renderer is processing commands, the return values of all **`glGet`*** functions may be affected by the current virtual screen. **`aglSetVirtualScreen`** may be used before a **`glGet`*** function to get values from a specific renderer.

ERRORS

aglSetVirtualScreen returns **GL_FALSE** if it fails for any reason.

AGL_BAD_CONTEXT is set if *ctx* is not a valid AGL context.

AGL_INVALID_VALUE is set if *screen* is not a valid virtual screen number.

SEE ALSO

aglChoosePixelFormat, **aglDescribePixelFormat**, **aglDevicesOfPixelFormat**, **aglGetVirtualScreen**, **aglNextPixelFormat**

aglSurfaceTexture

Texture from another context’s surface. **(New in AGL for Max OS X 10.2 Jaguar)**

C SPECIFICATION

```
#include <agl.h>
void aglSurfaceTexture ( AGLContext ctx,
   GLenum target,
   GLenum internalformat,
   AGLContext surfacecontext )
```

PARAMETERS

ctx	Specifies the AGL context.
target	Specifies an allowable 2D OpenGL texture target such as **GL_TEXTURE_2D** or **GL_TEXTURE_RECTANGLE_EXT**.
Internalformat	Specifies the internal texture layout, which must be a supported format listed on table 3.15, 3.16, 3.17 or 3.18 of the *OpenGL 1.3 Specification*.
surfacecontext	Specifies the AGL rendering context from which to get the texture.

DESCRIPTION

aglSurfaceTexture allows direct texturing from a surface by using the surface contents a the source data for the texture, behaving much the same way as **glTexImage2D**. The texture target, internal format must be supported the renderer of the target context. Additionally, the source surface geometry must be compatible with the texture target. Thus, if the texture target is **GL_TEXTURE_2D**, the surface must conform to power of two dimensions.

This routine is designed for performance so the graphics driver will attempt to provide an optimum data path, keeping the data in VRAM if possible.

ERRORS

aglSurfaceTexture sets errors as other OpenGL texturing functions do, which can be queried with **glGetError**.

SEE ALSO

glGetError, **glTexImage2D**

aglSwapBuffers

Exchanges front and back buffers.

C SPECIFICATION

```
#include <agl.h>
void aglSwapBuffers ( AGLContext ctx )
```

PARAMETERS

ctx	Specifies the AGL context.

DESCRIPTION

aglSwapBuffers exchanges the front and back buffers of the current drawable. The exchange either takes place during the vertical retrace of the monitor or immediately after **aglSwapBuffers** is called depending on the setting of **AGL_SWAP_INTERVAL**. All AGL rendering contexts share the same notion of which are front buffers and which are back buffers.

An implicit **glFlush** is done by **aglSwapBuffers** before it returns. Subsequent OpenGL commands can be issued immediately after calling **aglSwapBuffers**, but are not executed until the buffer exchange is completed. **aglSwapBuffers** normally returns immediately, but it will block if the caller tries to submit a second frame before the previously submitted frame is completed rendering. Another way to look at this is there can be only one frame "in flight" at a time. This behavior can be modified using **aglSetInteger** with the context option **AGL_SWAP_LIMIT** which controls whether swaps will block waiting for completion, see **aglSetInteger** for further details.

NOTES

The software renderer can be selected by calling **aglChoosePixelFormat** with the **AGL_RENDERER_ID** attribute set to **AGL_RENDERER_GENERIC_ID**.

ERRORS

AGL_BAD_CONTEXT is set if *ctx* is not a valid AGL context.

SEE ALSO

glFlush, **aglSetInteger**, **aglChoosePixelFormat**

aglUpdateContext

Notify context that the window geometry has changed.

C SPECIFICATION

```
#include <agl.h>
GLboolean aglUpdateContext ( AGLContext ctx )
```

PARAMETERS

ctx	Specifies the AGL context.

DESCRIPTION

`aglUpdateContext` must be called by the application any time the graphics port geometry has changed. It should be called after any drag, grow, or zoom action is performed on the window.

ERRORS

`aglUpdateContext` returns **`GL_FALSE`** if it fails for any reason, **`GL_TRUE`** otherwise.

`AGL_BAD_CONTEXT` is set if *ctx* is not a valid context.

`AGL_BAD_ALLOC` is set if a renderer is unable resize a buffer.

SEE ALSO

`aglSetDrawable`

aglUseFont

Create bitmap display lists from an Apple font.

C SPECIFICATION

```
#include <agl.h>
GLboolean aglUseFont ( AGLContext ctx,
   GLint fontID ,
   Style face ,
   GLint size ,
   GLint first ,
   GLint count ,
   GLint base )
```

PARAMETERS

ctx	Specifies the rendering context.
fontID	Specifies the font from which character glyphs are to be taken.
face	Specifies the font style.
size	Specifies the font size.
first	Specifies the index of the first glyph to be taken.
count	Specifies the number of glyph to be taken.
base	Specifies the index of the first display list to be generated.

DESCRIPTION

`aglUseFont` generates count display lists, named *base* through *base* +*count* -1, each containing a single **`glBitmap`** command. The parameters of the glBitmap command of display list *base* +i are derived from glyph *first* +i . Bitmap parameters *xorig* , *yorig* , *width* , and *height* are computed from font metrics as *zero* , *descent* -1, *font width* , and *ascent* +*descent* , respectively. *xmove* is taken from the glyph's width metric, and *ymove* is set to zero. Finally, the glyph's image is converted to the appropriate format for **`glBitmap`**.

Empty display lists are created for all glyphs that are requested and are not defined in font.

The currently defined fonts in `<fonts.h>` are as follows:

```
applFont        losAngeles

athens          monaco

cairo           sanFran

courier         times

geneva          symbol
```

`helvetica`	`systemFont`
`mobile`	`toronto`
`newYork`	`venice`
`london`	

To obtain a font number associated with a font name, use the **`GetFNum`** function. More details are listed in *Inside Macintosh* under Font Manager.

The currently defined font styles in the `Types.h` header file are as follows:

`normal`	`bold`
`italic`	`underline`
`outline`	`shadow`
`condense`	`extend`

The face may be the bitwise `OR` of any of the defined Mac OS font styles.

ERRORS

`aglUseFont` returns **`GL_FALSE`** if it fails, **`GL_TRUE`** otherwise.

`AGL_BAD_STATE` is set if the current AGL context is in display list construction mode.

`AGL_BAD_CONTEXT` is set if there is no current context.

SEE ALSO

`glBitmap`, **`glNewList`**

Glossary

2D Two-dimensional. See also planar.

3D Three-dimensional. See also spatial.

accelerator See graphics accelerator.

accumulation buffer A buffer in which multiple rendered frames can be composited to produce a single image.

aliasing The jagged edges (or staircasing) that result from drawing an image on a raster device such as a computer screen. Compare **anti-aliasing.**

alpha blending A process of using alpha information to create transparent objects.

alpha channel A color component in some color spaces whose value represents the opacity of the color defined in the other components. Compare **ARGB color structure.**

antialiasing The smoothing of jagged edges on a displayed shape by modifying the transparencies of individual pixels along the shape's edge. Compare **aliasing.**

API See **application programming interface.**

application programming interface (API) The total set of constants, data structures, routines, and other programming elements that allow developers to use some part of the system software.

Architecture Review Board (ARB) An independent consortium that controls the evolution of OpenGL. Permanent members currently include Digital Equipment Corporation, Evans and Sutherlin, Hewlett-Packard, IBM, Integraph, Intel, Microsoft, and Silicon Graphics.

B-spline curve A curve that passes smoothly through a series of control points.

bitmap A two-dimensional array of values, each of which represents the state of one pixel.

constant shading A method of shading surfaces in which the incident light color and intensity are calculated for a single point on a polygon and then applied to the entire polygon. Compare **Gouraud shading, Phong shading.**

culling Ignoring hidden image data to reduce the amount of time required to render a model.

depth buffer TBD.

display list A named list of OpenGL commands that can be precompiled for faster execution and possible reuse.

double buffering Building an image in an off-screen buffer prior to display which is used to provide smooth animation of objects.

feedback mode A mode in which OpenGL returns the processed geometric information (colors, pixel positions, and so on) to the application instead of rendering them into the frame buffer.

drawable An entity into which pixel data can be drawn, such as a window, a full-screen buffer, or an off-screen buffer.

frame buffer The buffer in which the final image is prepared and staged for display.

geometric primitive Any of the basic geometric objects defined by OpenGL in the GL library.

Gouraud shading A method of shading surfaces in which the incident light color and intensity are calculated for each vertex of a polygon and then interpolated linearly across the entire polygon. Compare **constant shading, Phong shading.**

graphics accelerator Any hardware device used to increase rendering speed.

image The two-dimensional product of rendering.

material lighting A process by which the color of a point on a surface is computed using the properties of the surface material.

modeling The process of creating a representation of real or abstract objects.

non-uniform rational B-spline (NURB or NURBS) A curve defined by non-uniform parametric ratios of B-spline polynomials. NURB curves can be used to define very complex curves and surfaces, as well as very common geometric objects (for instance, the conic sections).

NURB See **Non-Uniform Rational B-spline.**

NURB curve A three-dimensional curve represented by a NURB equation**.**

Phong shading A method of shading surfaces in which the incident light color and intensity are calculated for a series of points along each edge of a polygon and then interpolated across the entire polygon. Compare **constant shading, Gouraud shading.**

planar Contained completely in two dimensions (as, for example, a circle). See also **spatial.**

polygon A closed plane figure. See **general polygon, simple polygon.**

projection A method of mapping three-dimensional objects into two dimensions.

rasterization The process of determining values for the pixels in a rendered image. Also called scan conversion.

render To create an image (on the screen or some other medium) of a model.

renderer Software or firmware used to create an image from a view and a model.

rendering The process of creating an image (on the screen or some other medium) of a model. See also **rasterization.**

scale To reposition and resize an object by multiplying the x, y, and z coordinates of each of its points by values dx, dy, and dz.

simple polygon A closed plane figure defined by a list of vertices (that is, defined by a single contour).

stencil buffer A buffer used to mask individual pixels.

tessellate To decompose a curve or surface into polygonal faces.

texture mapping A technique wherein a predefined image (the texture) is mapped onto the surface of an object in a model.

transparency The ability of an object to allow light to pass through it.

vertex A dimensionless position in three- or four-dimensional space at which two or more lines (for instance, edges) intersect, with an optional set of vertex attributes

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh computers and FrameMaker software. Line art was created using Adobe™ Illustrator and Adobe Photoshop.

Text type is Palatino® and display type is Helvetica® . Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Courier New.

WRITERS
Robert Beretta, John Stauffer, Geoff Stahl

PUBLICATION ASSISTANCE
Michael Hinkson