OpenGL ES Framework Reference

Graphics & Animation: 3D Drawing

2009-06-11

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

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

The Apple logo is a trademark of Apple Inc.

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

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

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

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

Apple, the Apple logo, iPhone, Objective-C, and Pages are trademarks of Apple Inc., registered in the United States and other countries.

IOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used under license.

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

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

Introduction

Framework	/System/Library/Frameworks/OpenGLES.framework
Header file directories	/System/Library/Frameworks/OpenGLES.framework/Headers/
Companion guide	OpenGL ES Programming Guide for iOS
Declared in	EAGL.h EAGLDrawable.h

OpenGL ES provides a C-based interface used to accelerate 2D and 3D graphics rendering. The OpenGL ES framework (`OpenGLES.framework`) provided with the iOS provides implementations that conform to both the OpenGL ES 1.1 and OpenGL ES 2.0 specifications.

This collection of documents provides the reference for the platform specific APIs for OpenGL ES on the iPhone, also known as EAGL. EAGL provides graphics contexts that encapsulate all OpenGL ES state as the ability to configure a Core Animation layer to be the destination for OpenGL ES drawing commands. EAGL also allows OpenGL ES objects, such as textures, renderbuffers, and framebuffers, to be shared between two or more graphics contexts.

The Khronos Group maintains references for the OpenGL ES 1.1 and 2.0 programming interfaces:

- OpenGL ES API Registry is the official repository of OpenGL ES specification and extension documents provided by the Khronos Group.
- OpenGL ES 1.1 Reference Pages provides a complete reference to the OpenGL ES 1.1 specification, indexed alphabetically.
- OpenGL ES 2.0 Reference Pages provides a complete reference to the OpenGL ES 2.0 specification, indexed alphabetically.

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

Introduction

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

Classes

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

PART I

Classes

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

EAGLContext Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/OpenGLES.framework
Availability	Available in iOS 2.0 and later.
Declared in	EAGL.h EAGLDrawable.h
Companion guide	OpenGL ES Programming Guide for iOS
Related sample code	aurioTouch GLSprite SpeakHere

Overview

An `EAGLContext` object manages the state information, commands, and resources needed to draw using OpenGL ES. All OpenGL ES commands are executed in relation to an EAGL context.

Drawing resources such as textures and renderbuffers are managed for the `EAGLContext` object by an `EAGLSharegroup` object associated with the context. When a new `EAGLContext` object is initialized, you can choose to have it create a new `EAGLSharegroup` object or use one obtained from a previously created EAGL context.

To draw to an EAGL context, a complete framebuffer object must first be bound to the context. For more information on configuring rendering contexts, see *OpenGL ES Programming Guide for iOS*.

Tasks

Creating EAGL Contexts

`- initWithAPI:` (page 12)
Initializes and returns a newly allocated rendering context with the specified version of the OpenGL ES rendering API.

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

`- initWithAPI:sharegroup:` (page 13)
Initializes and returns a newly allocated rendering context with the specified version of OpenGL ES rendering API and the specified sharegroup.

Setting the Current EAGL Context

`+ setCurrentContext:` (page 11)
Makes the specified context the current rendering context for the calling thread.

Attaching Storage to a Renderbuffer

`- renderbufferStorage:fromDrawable:` (page 14)
Binds a drawable object's storage to an OpenGL ES renderbuffer object.

Displaying a Renderbuffer

`- presentRenderbuffer:` (page 13)
Displays a renderbuffer's contents on screen.

Getting EAGL Context Information

`API` (page 10) *property*
Specifies the OpenGL ES rendering API version supported by the receiver. (read-only)

`sharegroup` (page 11) *property*
The receiver's sharegroup object. (read-only)

`+ currentContext` (page 11)
Returns the current rendering context for the calling thread.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

API

Specifies the OpenGL ES rendering API version supported by the receiver. (read-only)

```
@property(readonly) EAGLRenderingAPI API
```

Availability
Available in iOS 2.0 and later.

Declared In
`EAGL.h`

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

sharegroup

The receiver's sharegroup object. (read-only)

```
@property(readonly) EAGLSharegroup *sharegroup
```

Discussion
You retrieve the sharegroup of a context when you want to create two or more contexts that share rendering resources. Call `initWithAPI:` (page 12) to initialize the first contex, retrieve its sharegroup, and then initialize additional contexts by calling `initWithAPI:sharegroup:` (page 13) passing this sharegroup as the parameter.

Availability
Available in iOS 2.0 and later.

See Also
`- initWithAPI:sharegroup:` (page 13)

Declared In
`EAGL.h`

Class Methods

currentContext

Returns the current rendering context for the calling thread.

```
+ (EAGLContext *)currentContext
```

Return Value
The current EAGL context for the calling thread.

Availability
Available in iOS 2.0 and later.

Declared In
`EAGL.h`

setCurrentContext:

Makes the specified context the current rendering context for the calling thread.

```
+ (BOOL)setCurrentContext:(EAGLContext *)context
```

Parameters
context
The rendering context that you want to make current.

Return Value
`YES` if successful; otherwise`NO`. If an error occurred, the rendering context for the current thread remains unchanged.

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

Discussion
All OpenGL ES calls are issued with respect to the current context and complete in the order they are called, unless otherwise specified.

EAGL retains the context when it is made current and releases the previous context. Calling this method with a `nil` parameter releases the current context and leaves OpenGL ES unbound to any drawing context.

You should avoid making the same context current on multiple threads. OpenGL ES provides no thread safety, so if you want to use the same context on multiple threads, you must employ some form of thread synchronization to prevent simultaneous access to the same context from multiple threads.

Availability
Available in iOS 2.0 and later.

Related Sample Code
aurioTouch
GLSprite
SpeakHere

Declared In
`EAGL.h`

Instance Methods

initWithAPI:

Initializes and returns a newly allocated rendering context with the specified version of the OpenGL ES rendering API.

```
- (id)initWithAPI:(EAGLRenderingAPI)api
```

Parameters

api

The desired version of the OpenGL ES rendering API. For legal values, see "OpenGL ES Versions" (page 15).

Return Value
An initialized context object or `nil` if the object couldn't be created.

Discussion
To issue OpenGL ES commands to this context, you must first make it the current drawing context by calling `setCurrentContext:` (page 11).

Calling `initWithAPI:` (page 12) creates a new `EAGLSharegroup` object and attaches it to this context.

Availability
Available in iOS 2.0 and later.

Related Sample Code
aurioTouch
GLSprite
SpeakHere

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

Declared In
EAGL.h

initWithAPI:sharegroup:

Initializes and returns a newly allocated rendering context with the specified version of OpenGL ES rendering API and the specified sharegroup.

```
- (id)initWithAPI:(EAGLRenderingAPI)api sharegroup:(EAGLSharegroup *)sharegroup
```

Parameters

api

The desired version of the OpenGL ES rendering API. For legal values, see "OpenGL ES Versions" (page 15).

sharegroup

A sharegroup obtained from another `EAGLContext` object.

Return Value
An initialized context object or `nil` if the object couldn't be created.

Discussion
To issue OpenGL ES commands to this context, you must first make it the current drawing context by calling `setCurrentContext:` (page 11).

OpenGL ES objects such as textures, renderbuffers, framebuffers and vertex buffers are shared across all contexts that are created with the same sharegroup. To specify that a new context should be initialized in an existing sharegroup, retrieve the `sharegroup` property from a previously initialized context and pass it as a parameter to this initialization method. If `nil` is passed as the `sharegroup` parameter, a new EAGLSharegroup (page 17) object is created and attached to the context.

Availability
Available in iOS 2.0 and later.

Declared In
EAGL.h

presentRenderbuffer:

Displays a renderbuffer's contents on screen.

```
- (BOOL)presentRenderbuffer:(NSUInteger)target
```

Parameters

target

The OpenGL ES binding point for a currently bound renderbuffer. For contexts that use the OpenGL ES 1.0 API, this must be `GL_RENDERBUFFER_OES`. For contexts that use the OpenGL ES 2.0 API, the OES suffix should be removed.

Return Value
`YES` if successful; otherwise `NO`.

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

Discussion
The renderbuffer to be displayed must have allocated storage using the `renderbufferStorage:fromDrawable:` (page 14) method. The exact semantics for how and when the renderbuffer contents are displayed is controlled by the drawable object.

Important: The contents of the renderbuffer may be altered after the renderbuffer is presented to the screen. After presenting the renderbuffer, your application must *completely* redraw the contents of the renderbuffer before presenting it again. To preserve the contents of the renderbuffer you may set the `kEAGLDrawablePropertyRetainedBacking` (page 22) key of the drawableProperties dictionary to `YES`. Setting the key to `YES` may result in reduced graphics performance and increased memory usage, so only do this when the contents of the renderbuffer must remain unchanged.

Availability
Available in iOS 2.0 and later.

See Also
`- renderbufferStorage:fromDrawable:` (page 14)

Related Sample Code
aurioTouch
GLSprite

Declared In
`EAGLDrawable.h`

renderbufferStorage:fromDrawable:

Binds a drawable object's storage to an OpenGL ES renderbuffer object.

```
- (BOOL)renderbufferStorage:(NSUInteger)target
    fromDrawable:(id<EAGLDrawable>)drawable
```

Parameters

target

The OpenGL ES binding point for a currently bound renderbuffer. For contexts that use the OpenGL ES 1.0 API, this must be `GL_RENDERBUFFER_OES`. For contexts that use the OpenGL ES 2.0 API, the OES suffix should be removed.

drawable

An object that conforms to the EAGLDrawable protocol whose storage will be bound to the renderbuffer. In iOS 3.0, this is always a `CAEAGLLayer` object.

Return Value
`YES` if successful; otherwise `NO`.

Discussion
To create a renderbuffer that can be presented to the screen, you bind the renderbuffer and then allocate shared storage for it by calling this method. This method call replaces the call normally made to `glRenderbufferStorage`. A renderbuffer whose storage has been allocated with this method can later be displayed with a call to `presentRenderbuffer:`.

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

The width, height, and internal color buffer format are derived from the characteristics of the drawable object. You may override the internal color buffer format by adding an `kEAGLDrawablePropertyColorFormat` (page 22) key to the `drawableProperties` dictionary of the drawable object before calling this method.

To specify that the OpenGL ES renderbuffer should be detached from the drawable object, you can call this method with `drawable` set to `nil`.

Availability
Available in iOS 2.0 and later.

Declared In
`EAGLDrawable.h`

Constants

OpenGL ES Versions

These constants are used to choose the version of OpenGL ES that a rendering context provides.

```
typedef NSUInteger EAGLRenderingAPI;
enum
{
    kEAGLRenderingAPIOpenGLES1          = 1
    kEAGLRenderingAPIOpenGLES2          = 2
};
```

Constants

`kEAGLRenderingAPIOpenGLES1`
Context supports OpenGL ES 1.x rendering API.
Available in iOS 2.0 and later.
Declared in `EAGL.h`.

`kEAGLRenderingAPIOpenGLES2`
Context supports OpenGL ES 2.x rendering API.
Available in iOS 3.0 and later.
Declared in `EAGL.h`.

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

EAGLSharegroup Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/OpenGLES.framework
Availability	Available in iOS 2.0 and later.
Declared in	EAGL.h
Companion guide	OpenGL ES Programming Guide for iOS

Overview

An `EAGLSharegroup` object manages OpenGL ES resources associated with one or more `EAGLContext` objects. It is created when an `EAGLContext` object is initialized and disposed of when the last `EAGLContext` object that references it is released. As an opaque object, there is no developer accessible API.

Currently, the sharegroup manages textures, buffers, framebuffers, and renderbuffers. It is your application's responsibility to manage state changes to shared objects when those objects are accessed from multiple contexts in the sharegroup. The results of changing the state of a shared object while it is being used for rendering in another context are undefined. To obtain deterministic results, your application must take explicit steps to ensure that the shared object is not currently being used for rendering while your application modifies it. Further, state changes are not guaranteed to be noticed by another context in the sharegroup until that context rebinds the shared object.

To ensure defined results of state changes to shared objects across contexts in the sharegroup, your application must perform the following tasks, in this order:

1. Call `glFlush` on the rendering context that issues the state-modifying routines.
2. Call `glBindTexture` or `glBindBuffer` on the rendering context that depends on the texture or vertex buffer object state changes, respectively.

A shared object is not deleted until it is no longer bound to any context.

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

Protocols

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

PART II

Protocols

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

EAGLDrawable Protocol Reference

Adopted by	CAEAGLLayer
Framework	/System/Library/Frameworks/OpenGLES.framework
Availability	Available in iOS 2.0 and later.
Declared in	EAGLDrawable.h
Companion guide	OpenGL ES Programming Guide for iOS

Overview

iOS objects that implement the `EAGLDrawable` protocol can be used as a rendering surface and displayed to the screen by an `EAGLContext` object. In iOS 2.0, this protocol is implemented only by the `CAEAGLLayer` class, but in the future other classes may choose to implement the protocol. The `EAGLDrawable` protocol is not intended to be implemented by objects outside of the iOS.

Tasks

Drawable Properties

`drawableProperties` (page 21) *required property*
A dictionary of values that specify the desired characteristics of the drawable surface. (required)

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

drawableProperties

A dictionary of values that specify the desired characteristics of the drawable surface. (required)

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

```
@property(copy) NSDictionary* drawableProperties;
```

Discussion
The `drawableProperties` dictionary specifies the properties that are used by this object when it is attached to an OpenGL ES renderbuffer. Your application should set these properties before passing this object into the `EAGLContext` method `renderbufferStorage:fromDrawable:` (page 14). If you change the `drawableProperties` dictionary, your application must call `renderbufferStorage:fromDrawable:` (page 14) again on the context for the new values to take effect.

Availability
Available in iOS 2.0 and later.

Related Sample Code
aurioTouch
GLSprite
SpeakHere

Declared In
`EAGLDrawable.h`

Constants

Drawable Property Keys

Keys to specify in the `drawableProperties` dictionary.

```
EAGL_EXTERN NSString * const kEAGLDrawablePropertyColorFormat;
EAGL_EXTERN NSString * const kEAGLDrawablePropertyRetainedBacking;
```

Constants

`kEAGLDrawablePropertyColorFormat`

The key specifying the internal color buffer format for the drawable surface. The value for this key is an `NSString` object that specifies a specific color buffer format. This color buffer format is used by the `EAGLContext` object to create the storage for a renderbuffer. The default value is `kEAGLColorFormatRGBA8`.

Available in iOS 2.0 and later.

Declared in `EAGLDrawable.h`.

`kEAGLDrawablePropertyRetainedBacking`

The key specifying whether the drawable surface retains its contents after displaying them. The value for this key is an `NSNumber` object containing a `BOOL` data type. If `NO`, you may not rely on the contents being the same after the contents are displayed. If `YES`, then the contents will not change after being displayed. Setting the value to `YES` is recommended only when you need the content to remain unchanged, as using it can result in both reduced performance and additional memory usage. The default value is `NO`.

Available in iOS 2.0 and later.

Declared in `EAGLDrawable.h`.

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

Color Formats

Color formats that can be specified under the `kEAGLDrawablePropertyColorFormat` key.

```
EAGL_EXTERN NSString * const kEAGLColorFormatRGB565;
EAGL_EXTERN NSString * const kEAGLColorFormatRGBA8;
```

Constants

`kEAGLColorFormatRGB565`

Specifies a 16-bit RGB format that corresponds to the OpenGL ES `GL_RGB565` format.

Available in iOS 2.0 and later.

Declared in `EAGLDrawable.h`.

`kEAGLColorFormatRGBA8`

Specifies a 32-bit RGBA format that corresponds to the OpenGL ES `GL_RGBA8888` format.

Available in iOS 2.0 and later.

Declared in `EAGLDrawable.h`.

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

Functions

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

PART III

Functions

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

EAGL Functions Reference

Framework: OpenGLES/EAGL.h

Declared in EAGL.h

Overview

This document describes the functions in the OpenGL ES framework.

Functions

EAGLGetVersion

Retrieves the version information for the EAGL implementation.

```
void EAGLGetVersion(
    unsigned int* major,
    unsigned int* minor);
```

Parameters

major

On output, the major version of the EAGL implementation.

minor

On output, the minor version of the EAGL implementation.

Discussion

If *major* and *minor* parameters are not `nil`, they return the major and minor version number of the EAGL implementation, respectively.

Availability

Available in iOS 2.0 and later.

Declared In

`EAGL.h`

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

Document Revision History

This table describes the changes to *OpenGL ES Framework Reference*.

Date	Notes
2009-06-11	Revised to include OpenGL ES 2.0 information.
2008-07-08	New document that describes the OpenGL ES programming interface, a high performance graphics library available on the iPhone.

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.

Document Revision History

2009-06-11 | © 2009 Apple Inc. All Rights Reserved.