OpenGL ES Framework Reference

Graphics & Animation: 3D Drawing

2008-07-08

Contents

Introduction

Framework	/System/Library/Frameworks/OpenGLES.framework
Header file directories	/System/Library/Frameworks/OpenGLES.framework/Headers/
Declared in	EAGL.h
	EAGLDrawable.h

This collection of documents provides the API reference for the OpenGL ES framework, which consists of OpenGL ES and EAGL. OpenGL ES provides a C-based interface used to accelerate 2D and 3D graphics rendering. The OpenGL ES framework (`OpenGLES.framework`) provided with iPhone OS conforms to the OpenGL ES v1.1 specification. You can find out more about OpenGL ES by reading Polygons In Your Pocket: Introducing OpenGL ES.

The EAGL API defines the interface between the iPhone OS and OpenGL ES. EAGL provides graphics contexts that encapsulate all OpenGL ES state and offers a mechanism to present rendered images to Core Animation for display. EAGL also allows OpenGL ES objects, such as textures, renderbuffers, and framebuffers, to be shared between two or more graphics contexts.

Classes

Classes

EAGLContext Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/OpenGLES.framework
Availability	Available in iPhone OS 2.0 and later.
Declared in	EAGL.h
	EAGLDrawable.h
Related sample code	aurioTouch
	GLGravity
	GLPaint
	GLSprite

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.

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.

– `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 10) *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 2.0 Programming Language*.

API

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

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

Availability
Available in iPhone OS 2.0 and later.

Declared In
`EAGL.h`

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 iPhone OS 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 iPhone OS 2.0 and later.

Related Sample Code
GLPaint

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.

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 iPhone OS 2.0 and later.

Related Sample Code

aurioTouch

GLGravity

GLPaint

GLSprite

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 EAGL 1.0, `api` must be kEAGLRenderingAPIOpenGLES1 (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 iPhone OS 2.0 and later.

Related Sample Code

aurioTouch

GLGravity

GLPaint

GLSprite

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 EAGL 1.0, api must be kEAGLRenderingAPIOpenGLES1 (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 iPhone OS 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. In EAGL 1.0, this must be GL_RENDERBUFFER_OES.

Return Value
YES if successful; otherwise NO.

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.

By default, the contents of the renderbuffer become undefined after calling this method. To ensure the
contents of the render buffer remain unchanged, you may set the
`kEAGLDrawablePropertyRetainedBacking` (page 22) key of the `drawableProperties` dictionary to
`YES`. Because setting the key to `YES` may result in reduced graphics performance and increased memory
usage, doing so is recommended only when you require the contents of the buffer to remain unchanged
after they are presented.

Availability

Available in iPhone OS 2.0 and later.

See Also

– `renderbufferStorage:fromDrawable:` (page 14)

Related Sample Code

aurioTouch

GLGravity

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. In EAGL 1.0, this must be
> `GL_RENDERBUFFER_OES`.

drawable

> An object that conforms to the EAGLDrawable protocol whose storage will be bound to the
> renderbuffer. In iPhone OS 2.0, this is a `CAEAGLLayer` object.

Return Value

`YES` if successful; otherwise `NO`.

Discussion

To create a renderbuffer that can be presented to the screen, you bind a renderbuffer using
`glBindRenderbufferOES` 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:`

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 iPhone OS 2.0 and later.

Declared In
`EAGLDrawable.h`

Constants

EAGLRenderingAPI

Supported OpenGL ES versions.

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

Constants
`kEAGLRenderingAPIOpenGLES1`
> Context supports OpenGL ES 1.x rendering API.
>
> Available in iPhone OS 2.0 and later.
>
> Declared in `EAGL.h`.

Availability
Available in iPhone OS 2.0 and later.

Declared In
`EAGL.h`

EAGLSharegroup Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/OpenGLES.framework
Availability	Available in iPhone OS 2.0 and later.
Declared in	EAGL.h

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.

Protocols

Protocols

EAGLDrawable Protocol Reference

Adopted by	CAEAGLLayer
Conforms to	
Framework	/System/Library/Frameworks/OpenGLES.framework
Availability	Available in iPhone OS 2.0 and later.
Declared in	EAGLDrawable.h

Overview

iPhone OS objects that implement the `EAGLDrawable` protocol can be used as a rendering surface and displayed to the screen by an `EAGLContext` object. In iPhone OS 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 iPhone OS.

Tasks

Drawable Properties

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

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C 2.0 Programming Language*.

drawableProperties

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

```
@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 iPhone OS 2.0 and later.

Related Sample Code

aurioTouch

GLGravity

GLPaint

GLSprite

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 iPhone OS 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 iPhone OS 2.0 and later.

> Declared in `EAGLDrawable.h`.

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 iPhone OS 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 iPhone OS 2.0 and later.

Declared in `EAGLDrawable.h`.

Functions

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 iPhone OS 2.0 and later.

Declared In

`EAGL.h`

Document Revision History

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

Date	Notes
2008-07-08	New document that describes the OpenGL ES programming interface, a high performance graphics library available on the iPhone.

Index