QCPlugInOutputImageProvider Protocol Reference

Cocoa > Graphics & Imaging

2007-05-09

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

Simultaneously published in the United States and Canada.

Contents

QCPlugInOutputImageProvider Protocol Reference

Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	QCPlugIn.h

Overview

The `QCPlugInOuputImageProvider` protocol eliminates the need to use explicit image types for the image output ports on a custom patch. The methods in this protocol are called by the Quartz Composer engine when the output image is needed. If your custom patch has an image output port, you need to implement the appropriate methods for rendering image data and to supply information about the rendering destination and the image bounds.

Output images are opaque provider objects that comply to this protocol. To create an image output port as an Objective-C 2.0 property, declare it as follows:

```
@property(dynamic) id<QCPlugInOutputImageProvider> outputImage;
```

To create an image input port dynamically use the type `QCPortTypeImage`:

```
[self addOutputPortWithType:QCPortTypeImage
                     forKey:@"outputImage"
             withAttributes:nil];
```

To write images to that port, you need to implement the methods in this protocol and create an internal class that represents the images produced by the custom patch. For example, a simple interface for an image provider is:

```
@interface MyOutputImage : NSObject <QCPlugInOutputImageProvider>
{
    NSUInteger _width;
    NSUInteger _height;
}
```

Tasks

Rendering an Image to a Destination

`- renderToBuffer:withBytesPerRow:pixelFormat:forBounds:` (page 8)
Renders a subregion of the image into the supplied memory buffer using the specified pixel format.

- `copyRenderedTextureForCGLContext:pixelFormat:bounds:isFlipped:` (page 7)
  Returns the name of an OpenGL texture of type `GL_TEXTURE_RECTANGLE_EXT` that contains a subregion of the image in a given pixel format.
- `renderWithCGLContext:forBounds:` (page 9)
  Renders a subregion of the image to the provided CGL context.
- `releaseRenderedTexture:forCGLContext:` (page 8)
  Releases the previously copied texture.

Providing Information About the Image

- `imageBounds` (page 7)
  Returns the bounds of the image expressed in pixels and aligned to integer boundaries.
- `imageColorSpace` (page 8)
  Returns the color space of the image or `NULL` if the image should not be color matched.
- `shouldColorMatch` (page 10)
  Returns whether the image should be color matched.

Providing Information About the Rendering Destination

- `supportedBufferPixelFormats` (page 10)
  Returns a list of pixel formats that are supported for rendering to a memory buffer.
- `supportedRenderedTexturePixelFormats` (page 10)
  Returns a list of pixel formats that are supported for rendering to an onscreen OpenGL context.
- `canRenderWithCGLContext:` (page 6)
  Returns whether the image data can be rendered into the provided CGL context.

Instance Methods

canRenderWithCGLContext:

Returns whether the image data can be rendered into the provided CGL context.

```
- (BOOL) canRenderWithCGLContext:(CGLContextObj)cgl_ctx
```

Parameters

ctx
  The CGL context that your image will be rendered to.

Return Value
`YES` if the image can be rendered into this CGL context; otherwise `NO`, in which case `renderToBuffer:withBytesPerRow:pixelFormat:forBounds:` (page 8) is called.

Discussion
If your image can render using any OpenGL context, simply return `YES`. If your code requires special extensions, you'll need to check for them and then provide the appropriate return value. For more information on checking for OpenGL capabilities supported by the hardware, see *OpenGL Programming Guide for Mac OS X*.

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

Declared In
`QCPlugIn.h`

copyRenderedTextureForCGLContext:pixelFormat:bounds:isFlipped:

Returns the name of an OpenGL texture of type `GL_TEXTURE_RECTANGLE_EXT` that contains a subregion of the image in a given pixel format.

```
- (GLuint) copyRenderedTextureForCGLContext:(CGLContextObj)cgl_ctx
    pixelFormat:(NSString*)format bounds:(NSRect)bounds isFlipped:(BOOL*)flipped
```

Parameters

cgl_ctx
The CGL context to render to.

format
A string that represents the pixel format of the texture.

bounds
The bounds of the subregion of the image.

isFlipped
Set to `YES` on output if the contents of the returned texture are vertically flipped.

Return Value
The name of an OpenGL texture of type `GL_TEXTURE_RECTANGLE_EXT` that contains a subregion of the image in a given pixel format or `0` if the texture can't be provided.

Discussion
Implement this method if you want to create the texture yourself or use framebuffer objects (FBO). Use `<OpenGL/CGLMacro.h>` to send commands to the OpenGL context. Make sure to preserve all the OpenGL states except the ones defined by `GL_CURRENT_BIT`.

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

Declared In
`QCPlugIn.h`

imageBounds

Returns the bounds of the image expressed in pixels and aligned to integer boundaries.

```
- (NSRect) imageBounds;
```

Return Value
The bounds of the image. Note that the `QCPlugIn` class does not support images that have infinite bounds.

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

Declared In
`QCPlugIn.h`

imageColorSpace

Returns the color space of the image or `NULL` if the image should not be color matched.

```
- (CGColorSpaceRef) imageColorSpace
```

Return Value
The color space of the image or `NULL`.

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

Declared In
`QCPlugIn.h`

releaseRenderedTexture:forCGLContext:

Releases the previously copied texture.

```
- (void) releaseRenderedTexture:(GLuint)name forCGLContext:(CGLContextObj)cgl_ctx;
```

Parameters

name
The name of the previously bound texture.

cgl_ctx
The CGL context.

Discussion
Your OpenGL code should save and restore all states *except* for those that are part of `GL_CURRENT_BIT` (vertex position, color, texture, and so on). Also use CGL macros instead of changing the current context, by including this statement:

```
#import <OpenGL/CGLMacro.h>
```

For more details, see *Quartz Composer Custom Patch Programming Guide*.

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

Declared In
`QCPlugIn.h`

renderToBuffer:withBytesPerRow:pixelFormat:forBounds:

Renders a subregion of the image into the supplied memory buffer using the specified pixel format.

```
- (BOOL) renderToBuffer:(void*)baseAddress withBytesPerRow:(NSUInteger)rowBytes
    pixelFormat:(NSString*)format forBounds:(NSRect)bounds
```

Parameters

baseAddress

The base address of the memory buffer. The Quartz Composer engine passes you an address that is aligned on a 16-byte boundary.

rowBytes

The number of bytes per row of the image data. The Quartz Composer engine guarantees this value is a multiple of 16.

format

The pixel format of the image data.

bounds

The bounds of the subregion.

Return Value

`YES` if the image is rendered successfully into the buffer; `NO` on failure or if the image provider doesn't support CPU rendering.

Discussion

The Quartz Composer engine calls this method when it needs pixels. It gives you the base address, the number of row bytes, and the format. Then, you write pixels to the buffer.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- renderWithCGLContext:forBounds:` (page 9)

Declared In

`QCPlugIn.h`

renderWithCGLContext:forBounds:

Renders a subregion of the image to the provided CGL context.

```
- (BOOL) renderWithCGLContext:(CGLContextObj)cgl_ctx forBounds:(NSRect)bounds
```

Parameters

cgl_ctx

The CGL context to render to.

bounds

The bounds of the subregion.

Return Value

`YES` if successful; `NO` on failure or if the image provider doesn't support GPU rendering.

Discussion

The view port is set for you. The model view and projection matrixes are set to the identity.

Your OpenGL code should save and restore all states *except* for those that are part of `GL_CURRENT_BIT` (vertex position, color, texture, and so on). Also use CGL macros instead of changing the current context, by including this statement:

```
#import <OpenGL/CGLMacro.h>
```

For more details, see *Quartz Composer Custom Patch Programming Guide*.

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

See Also
`- renderToBuffer:withBytesPerRow:pixelFormat:forBounds:` (page 8)

Declared In
`QCPlugIn.h`

shouldColorMatch

Returns whether the image should be color matched.

```
- (BOOL) shouldColorMatch
```

Return Value
`NO` if the image is a mask or gradient; otherwise `YES`, which is the default.

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

Declared In
`QCPlugIn.h`

supportedBufferPixelFormats

Returns a list of pixel formats that are supported for rendering to a memory buffer.

```
- (NSArray*) supportedBufferPixelFormats
```

Return Value
A list of pixel formats, in order of preference, that the image can be rendered to in memory, or `nil` if the image provider does not support rendering to the CPU.

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

See Also
`- supportedRenderedTexturePixelFormats` (page 10)

Declared In
`QCPlugIn.h`

supportedRenderedTexturePixelFormats

Returns a list of pixel formats that are supported for rendering to an onscreen OpenGL context.

```
- (NSArray*) supportedRenderedTexturePixelFormats
```

Return Value
Returns the list of texture pixel formats supported by `copyRenderedTextureForCGLContext:pixelFormat:bounds:isFlipped:` (page 7) or `nil` if not supported.

Discussion
If this method returns nil, then Quartz Composer calls `canRenderWithCGLContext:` (page 6) /`renderWithCGLContext:forBounds:` (page 9).

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

See Also
`- supportedBufferPixelFormats` (page 10)

Declared In
`QCPlugIn.h`

Document Revision History

This table describes the changes to *QCPlugInOutputImageProvider Protocol Reference*.

Date	Notes
2007-05-09	New document that describes the methods for managing image data that's produced as output from a QCPlugIn object.

Document Revision History

Index