OpenGL ES Overview for iPhone OS

Gokhan Avkarogullari
iPhone GPU Software

Richard Schreyer
iPhone GPU Software


OpenGL|ES™
2.0

New Extensions

Retina Display

Large Display Performance

Multitasking

New Extensions

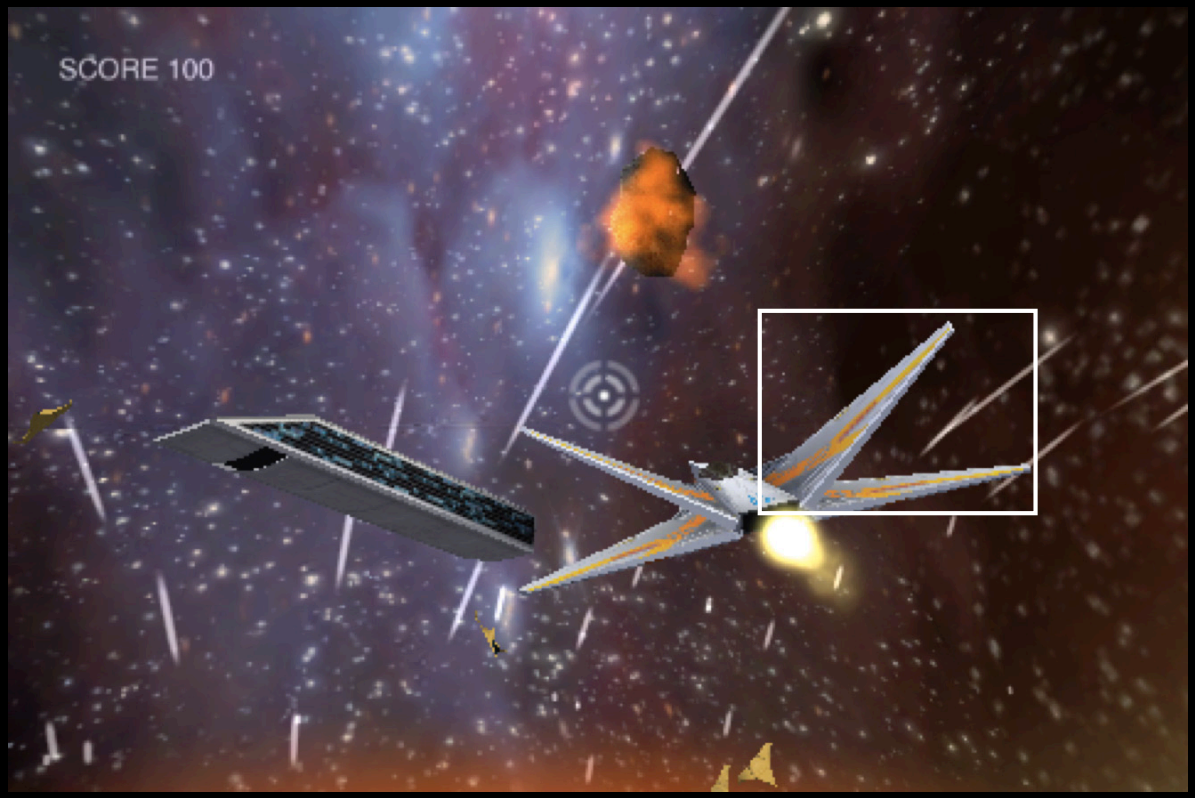
Retina Display

Large Display Performance

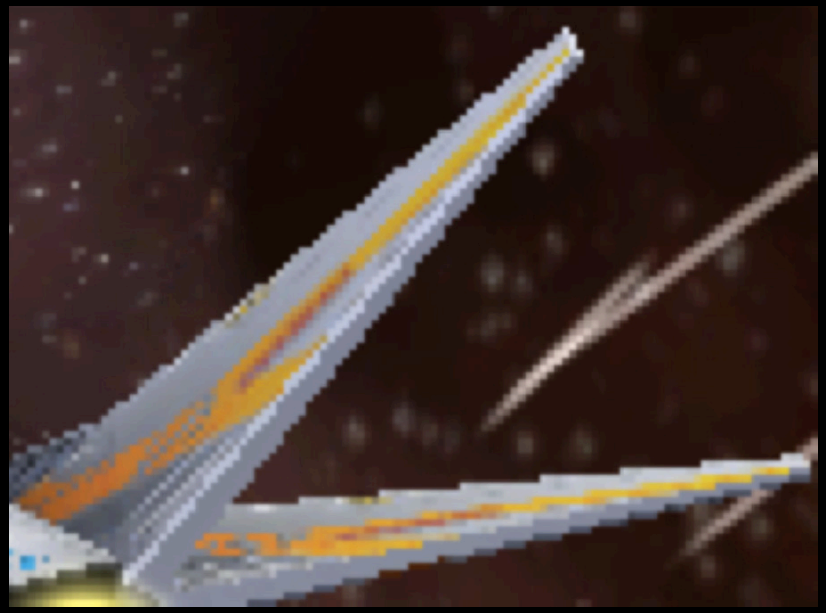
Multitasking

Multisample Framebuffer

Aliasing

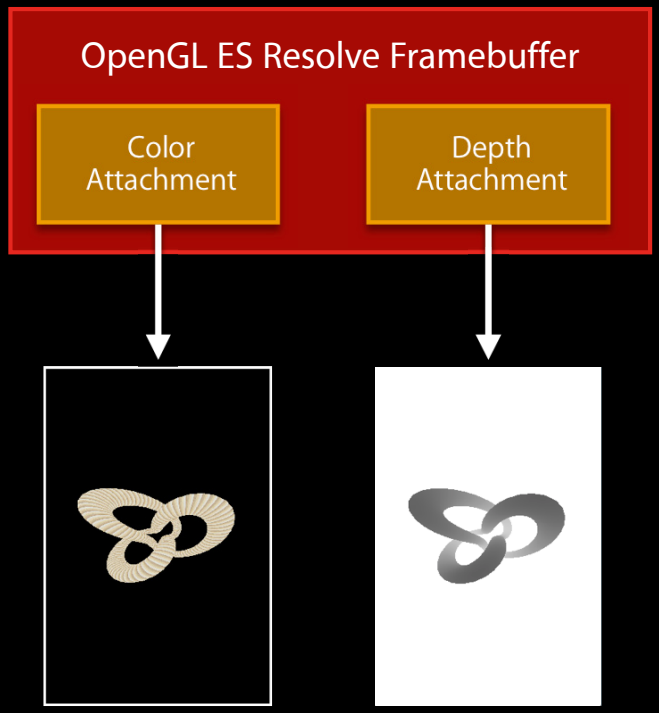


Anti-Aliasing



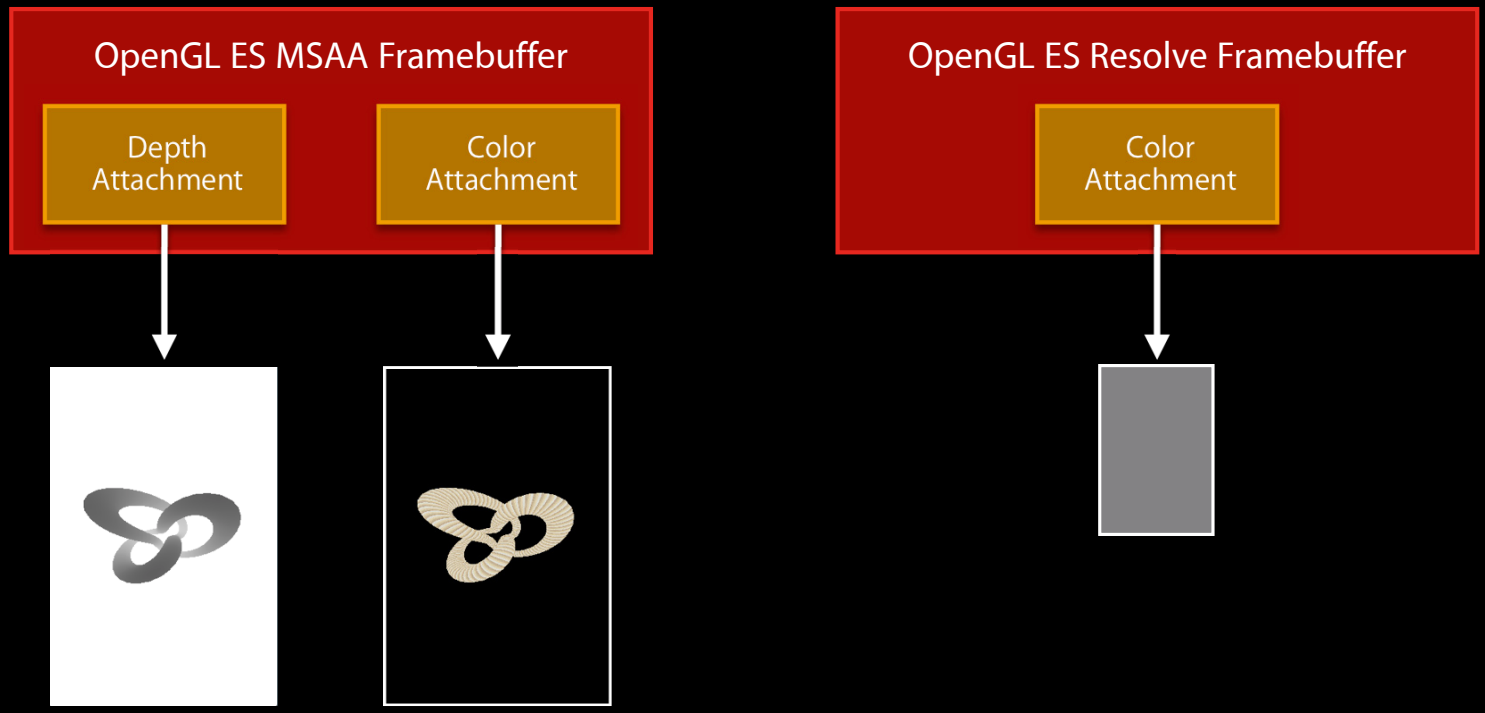
Anti-Aliasing via Multisampling

Framebuffer object for single sampling



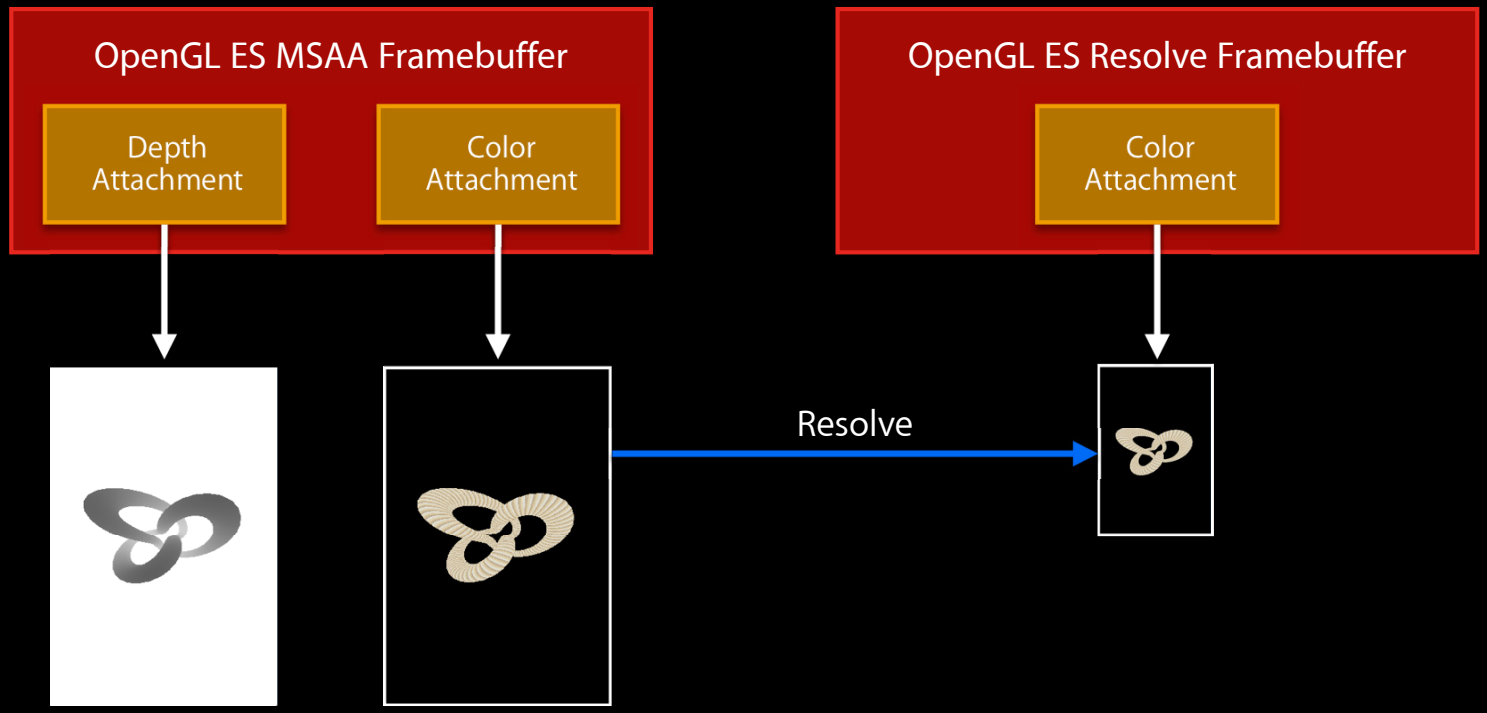
Anti-Aliasing via Multisampling

Framebuffer objects for multisampling



Anti-Aliasing via Multisampling

Framebuffer objects for multisample and resolve



Anti-Aliasing

Single sampled framebuffer creation

```
//color renderbuffer
   glGenRenderbuffers(1, &colorRenderbuffer);
   glBindRenderbuffer(GL_RENDERBUFFER, colorRenderbuffer);
   [context renderbufferStorage:GL_RENDERBUFFER fromDrawable:layer];
   glFramebufferRenderbuffer(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, ...);


//depth renderbuffer
   glGenRenderbuffers(1, &depthbuffer);
   glBindRenderbuffer(GL_RENDERBUFFER, depthbuffer);
   glRenderbufferStorage(GL_RENDERBUFFER, ..., );
   glFramebufferRenderbuffer(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT, ...);
```

Anti-Aliasing

Multisampled framebuffer creation

```
//color renderbuffer
   glGenRenderbuffers(1, &colorRenderbuffer);
   glBindRenderbuffer(GL_RENDERBUFFER, colorRenderbuffer);
   glRenderbufferStorageMultisampleAPPLE(GL_RENDERBUFFER_OES, samplesToUse, ...);
   glFramebufferRenderbuffer(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, ...);


//depth renderbuffer
   glGenRenderbuffers(1, &depthbuffer);
   glBindRenderbuffer(GL_RENDERBUFFER, depthbuffer);
   glRenderbufferStorageMultisampleAPPLE(GL_RENDERBUFFER, samplesToUse, ...);
   glFramebufferRenderbuffer(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT, ...);
```

Anti-Aliasing

Single sampled render and resolve

```
glBindFramebuffer(GL_FRAMEBUFFER, defaultFramebuffer);
glViewport(0, 0, backingWidth, backingHeight);

.... render

glBindRenderbufferOES(GL_RENDERBUFFER_OES, colorRenderbuffer);
[context presentRenderbuffer:GL_RENDERBUFFER_OES];
```

Anti-Aliasing

Multisampled render and resolve

```
glBindFramebuffer(GL_FRAMEBUFFER, msaaFramebuffer);
glViewport(0, 0, backingWidth, backingHeight);

.... render

glBindFramebuffer(GL_READ_FRAMEBUFFER_APPLE, msaaFramebuffer);
glBindFramebuffer(GL_DRAW_FRAMEBUFFER_APPLE, defaultFramebuffer);
glResolveMultisampleFramebufferAPPLE();

glBindRenderbufferOES(GL_RENDERBUFFER_OES, colorRenderbuffer);
[context presentRenderbuffer:GL_RENDERBUFFER_OES];
```

Anti-Aliasing

Implementation details

- PowerVR SGX supports hardware-accelerated multisampling
  - Multiple samples for each pixel
  - Runs shader once
    - Computes single color value for the entire fragment
  - Computes depth for each sample
    - If depth test passes, stores color for that sample
  - Averages sample colors to final pixel color during resolve

Anti-Aliasing

Implementation details

- On PowerVR MBX Lite, implemented as super sampling
  - Scene is rendered to a large buffer
  - Pixel values are computed for each sample
    - Then resolved to a single sampled buffer
  - Performance trade-off
    - Better than render-to-texture
      - Especially when discard extension is used

Demo

New Extensions

OpenGL ES in iOS 4

- `APPLE_framebuffer_multisample`

Discard Framebuffer

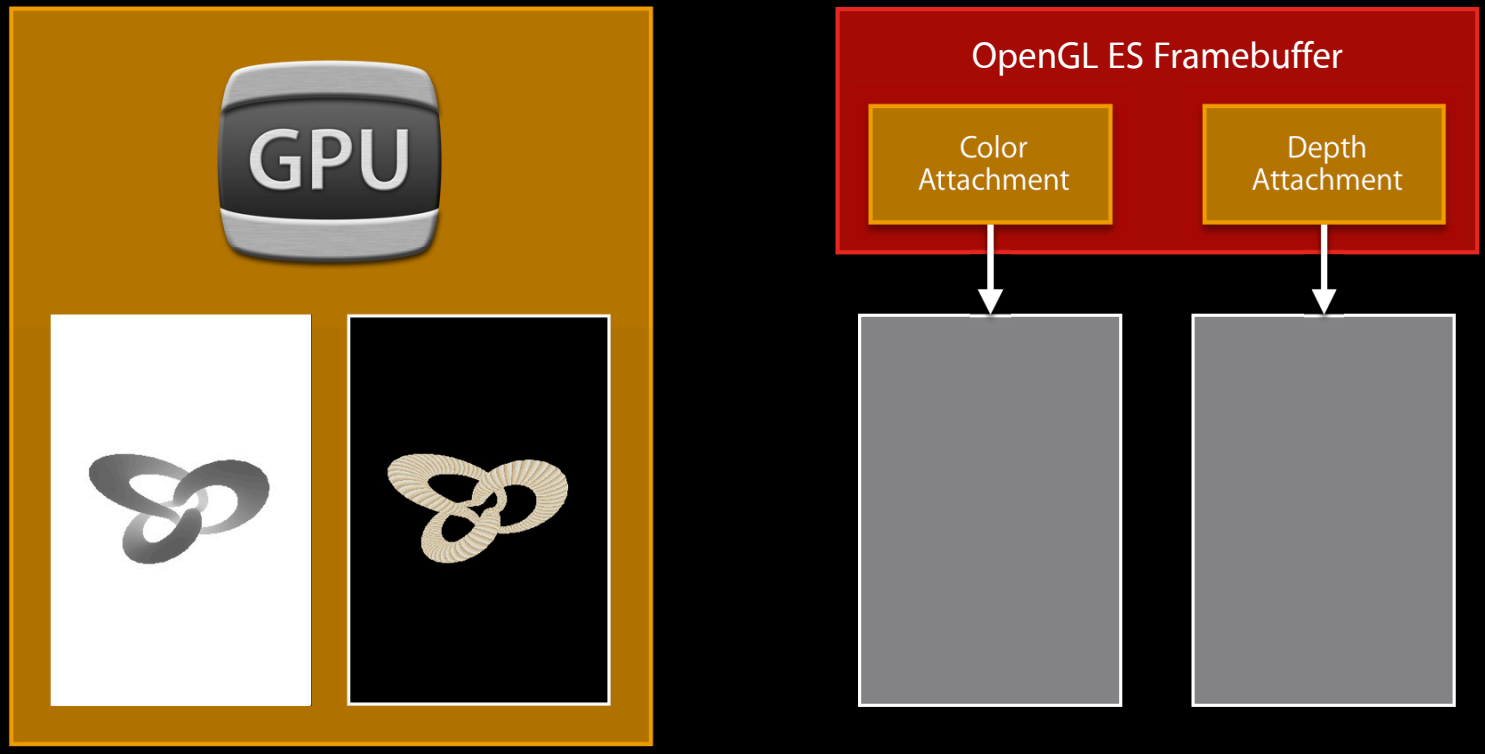
Discard Framebuffer

EXT_discard_framebuffer

- After the scene is rendered:
  - Depth buffer contents no longer needed
  - Stencil buffer contents no longer needed
  - Multisample color/depth buffers contents no longer needed
- By "discarding"
  - Avoids writing to depth/stencil and multisampled buffers
  - Reduces memory bandwidth usage
  - Enhances performance, especially for MSAA

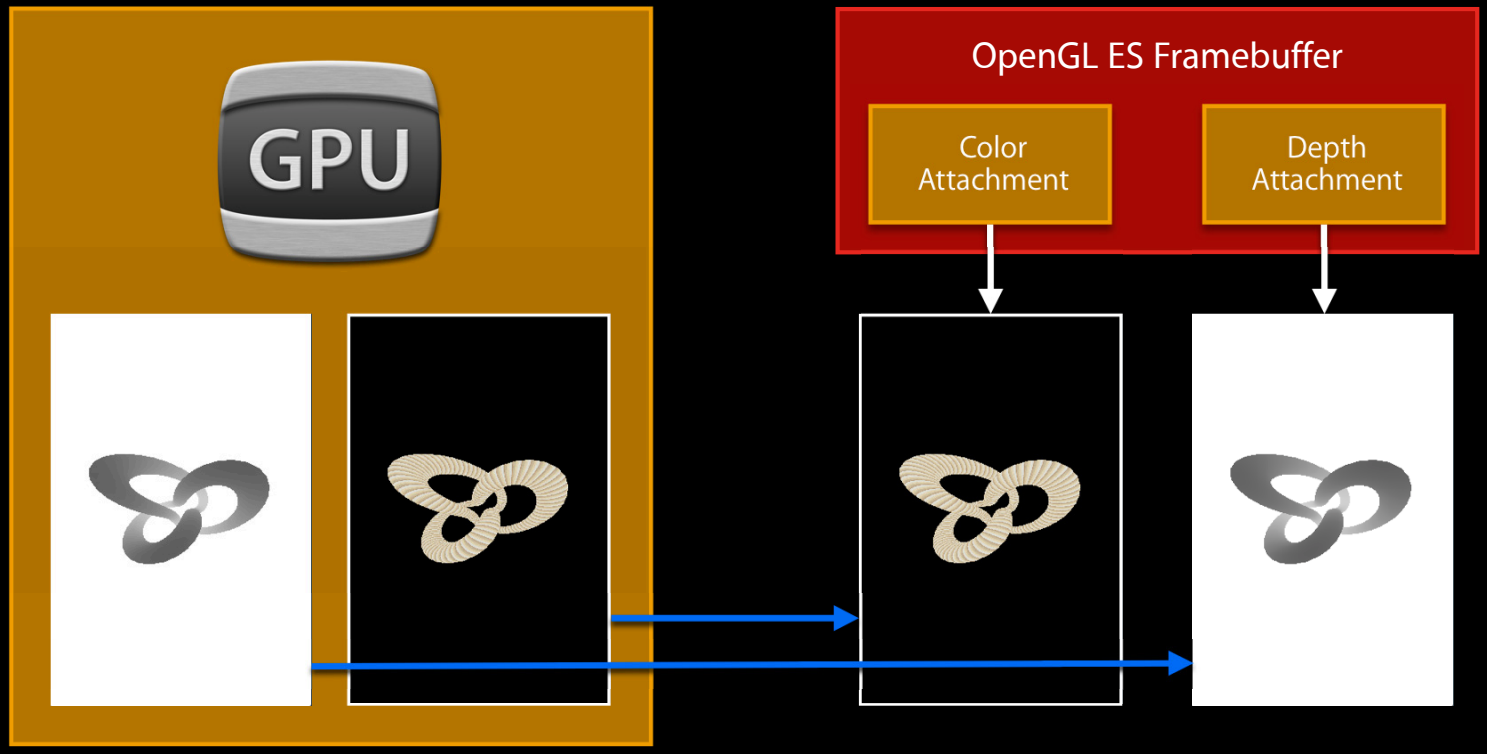
Discard Framebuffer

EXT_discard_framebuffer—before writing out



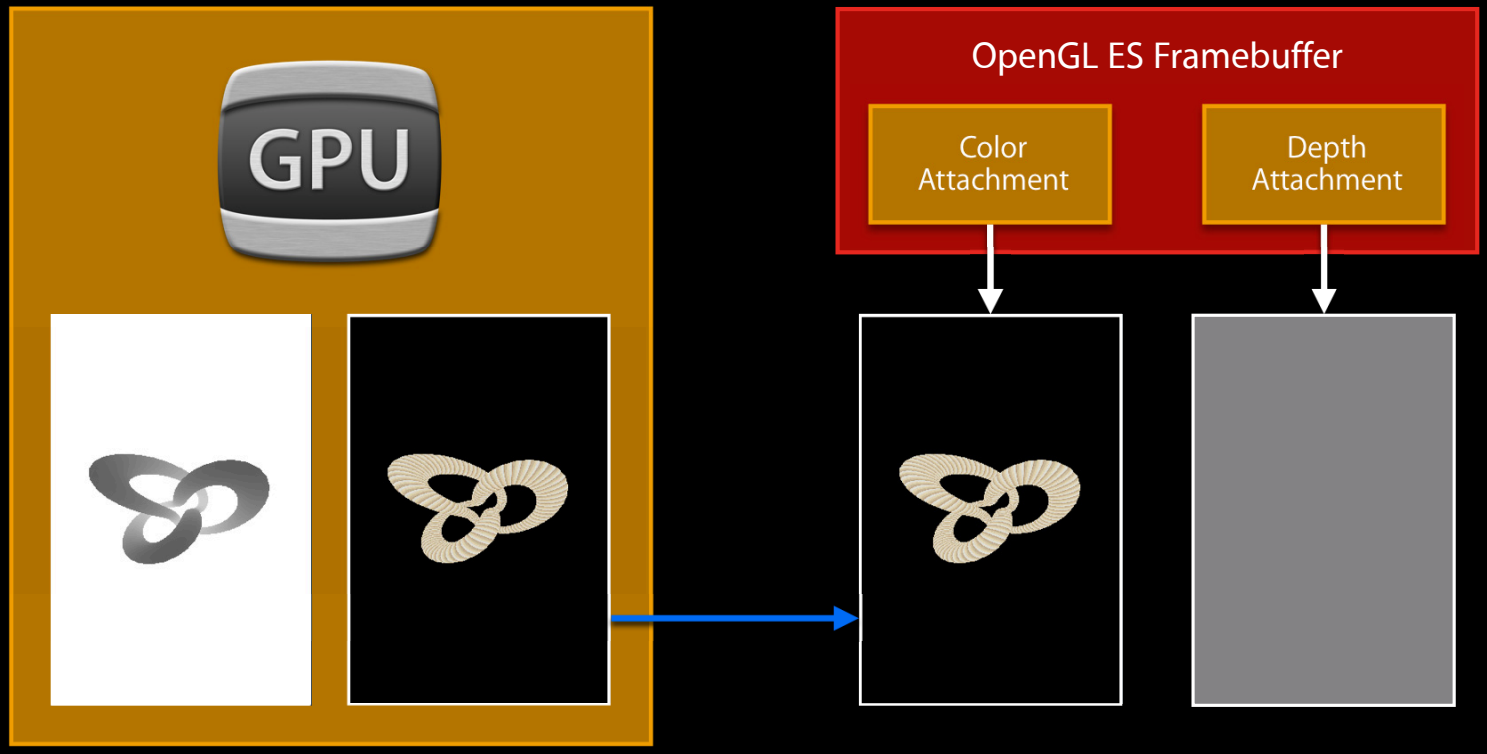
Discard Framebuffer

EXT_discard_framebuffer—without discard



Discard Framebuffer

EXT_discard_framebuffer—with discard



Discard Framebuffer

Code sample—without discard

```
glBindFramebuffer(GL_FRAMEBUFFER, defaultFramebuffer);
glViewport(0, 0, backingWidth, backingHeight);

.... render

glBindRenderbufferOES(GL_RENDERBUFFER_OES, colorRenderbuffer);
[context presentRenderbuffer:GL_RENDERBUFFER_OES];
```

Discard Framebuffer

Code sample—with discard

```
glBindFramebuffer(GL_FRAMEBUFFER, defaultFramebuffer);
glViewport(0, 0, backingWidth, backingHeight);

.... render

GLenum attachments[] = {GL_DEPTH_ATTACHMENT};
glDiscardFramebufferEXT(GL_READ_FRAMEBUFFER_APPLE, 1, attachments);

glBindRenderbufferOES(GL_RENDERBUFFER_OES, colorRenderbuffer);
[context presentRenderbuffer:GL_RENDERBUFFER_OES];
```

New Extensions

OpenGL ES in iOS 4

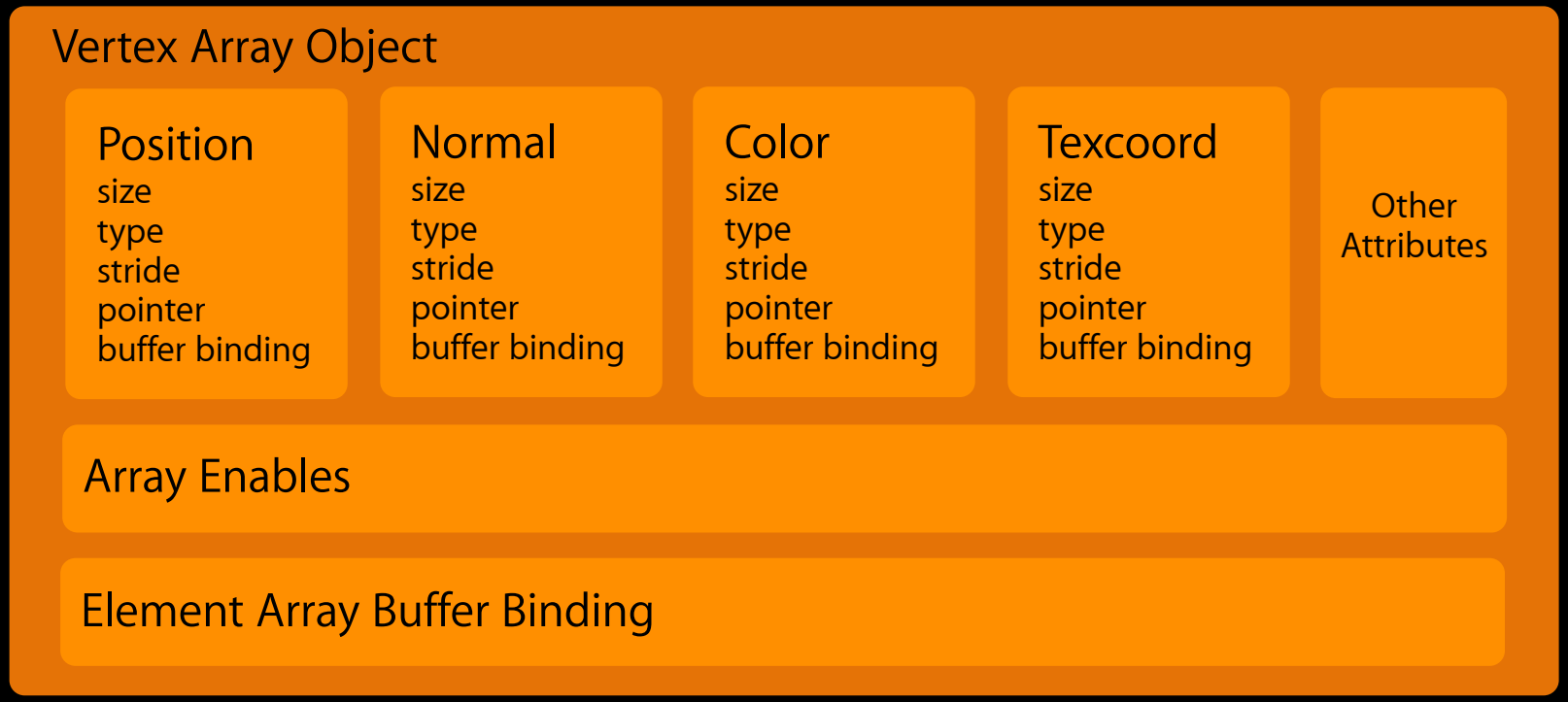
- APPLE_framebuffer_multisample
- EXT_discard_framebuffer

Vertex Array Objects

Vertex Array Objects

What are they?

- Encapsulate all states for vertex submission into a single object
- Bind between different vertex array configurations in a single call



Vertex Array Objects

Benefits

- Convenience
  - Array definition linked with asset loading
  - Fewer drawing calls
- Optimization
  - Makes certain vertex submission cases faster
    - Static vertex data
    - Indexed triangles
    - Others
  - Less state validation

Vertex Array Objects

Example

```
// Every draw call
// Set up vertex arrays
   glBindBuffer(GL_ARRAY_BUFFER, myBuffers[0]);
   glVertexPointer(3, GL_FLOAT, ...);
   glNormalPointer(GL_FLOAT, ...);
   glTexCoordPointer(2, GL_FLOAT, ...);

// Set up non-default enables
   glEnableClientState(GL_NORMAL_ARRAY);
   glEnableClientState(GL_TEXTURE_COORD_ARRAY);

// Draw
   glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, myBuffers[1]);
   glDrawArrays(GL_TRIANGLE_STRIP, ...);

// Restore non-default enables
   glDisableClientState(GL_NORMAL_ARRAY);
   glDisableClientState(GL_TEXTURE_COORD_ARRAY);
```

Vertex Array Objects

Example

```
// Every draw call
```

```
// Draw
   glBindVertexArrayOES(myVertexArray);
   glDrawArrays(GL_TRIANGLE_STRIP, ...);
```

Vertex Array Objects

Example

```
// One time setup
   glBindVertexArrayOES(myVertexArray);
   glVertexPointer(3, GL_FLOAT, ...);
   glNormalPointer(GL_FLOAT, ...);
   glTexCoordPointer(2, GL_FLOAT, ...);

   glEnableClientState(GL_VERTEX_ARRAY);
   glEnableClientState(GL_NORMAL_ARRAY);
   glEnableClientState(GL_TEXTURE_COORD_ARRAY);

// Every draw call
// Draw
   glBindVertexArrayOES(myVertexArray);
   glDrawArrays(GL_TRIANGLE_STRIP, ...);
```

New Extensions

OpenGL ES in iOS 4

- APPLE_framebuffer_multisample
- EXT_discard_framebuffer
- OES_vertex_array_object

More Extensions

Improved Texture Atlas Support

APPLE_texture_max_level

- Application specifies the maximum (coarsest) mipmap level that may be selected for the specified texture
- Controls filtering across atlas boundaries
- Reduces visual artifacts and improves performance

Improved Texture Atlas Support

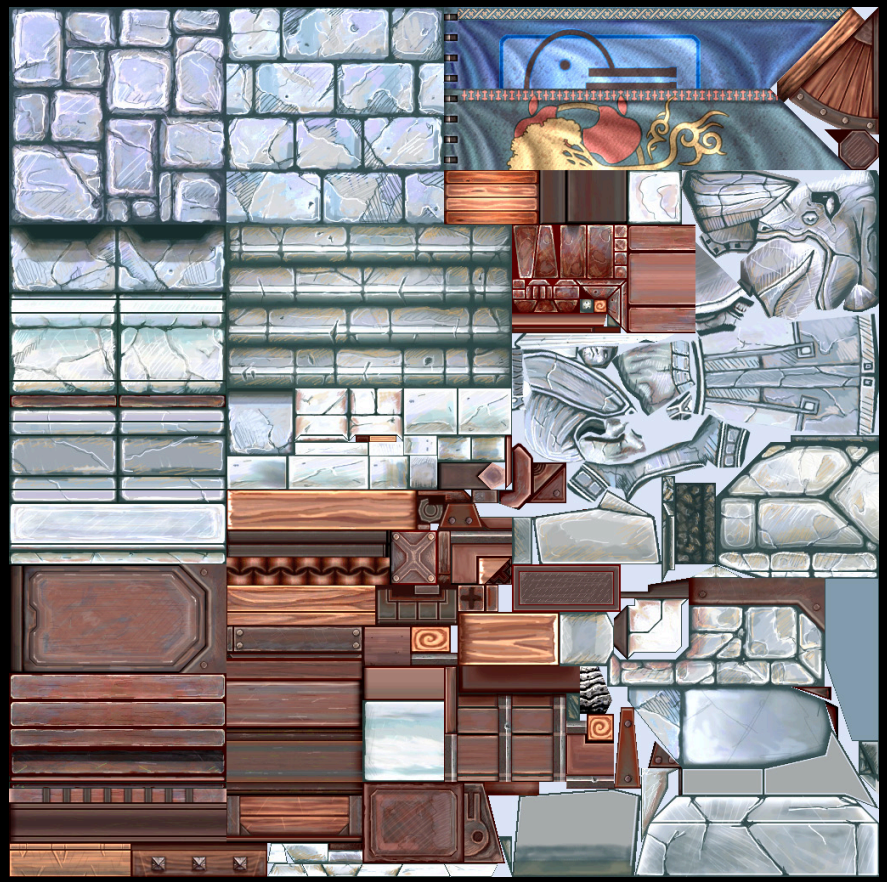
APPLE_texture_max_level

- Application specifies the maximum (coarsest) mipmap level that may be selected for the specified texture
- Controls filtering across atlas boundaries
- Reduces visual artifacts and improves performance

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAX_LEVEL_APPLE, maxLevel);
```

Improved Texture Atlas Support

APPLE_texture_max_level—texture atlas



Improved Texture Atlas Support

APPLE_texture_max_level—mipmaps in pixels

Improved Texture Atlas Support

APPLE_texture_max_level—mipmaps in coordinates

Improved Texture Atlas Support

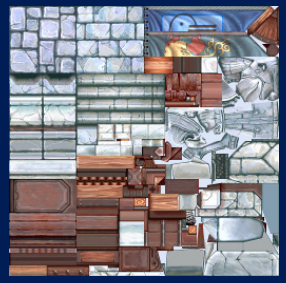
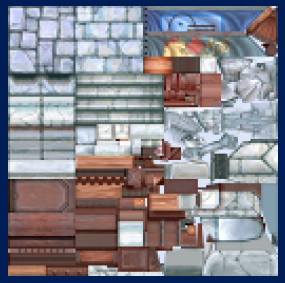
APPLE_texture_max_level—mipmaps in coordinates

Improved Texture Atlas Support

APPLE_texture_max_level—mipmaps in coordinates

Improved Texture Atlas Support

APPLE_texture_max_level—mipmaps with max level


Shader Texture LOD

APPLE_shader_texture_lod

- Provides explicit control of level of detail (LOD) in shaders
  - Gives control over which mipmap level is used
    - Can use finer mipmap level to sharpen the look of the object
    - Or use coarser mipmap level to reduce memory bandwidth usage

Shader Texture LOD

APPLE_shader_texture_lod

- Provides explicit control of level of detail at shaders
  - Gives control over which mipmap level is used
    - Finer mipmap levels take longer to read from disk
    - Allows rendering using coarser mipmap levels
- In your shader:
  - Enable

```
#extension GL_EXT_shader_texture_lod : enable
```

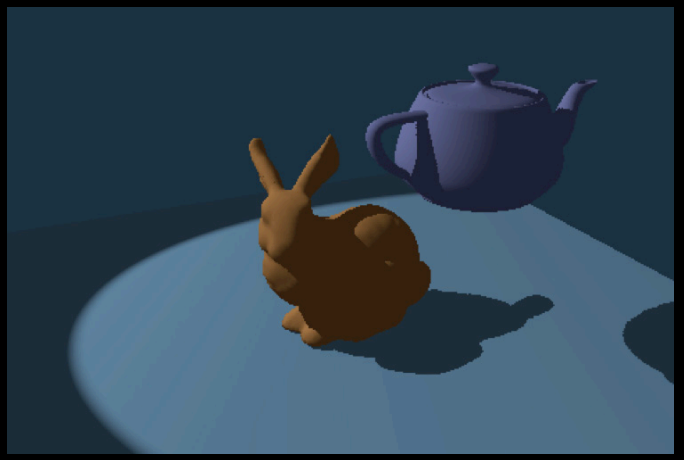
  - Control the mipmap level

```
texColor = texture2DLod(sampler2, test_vec2 , lod );
```

Depth Texture

OES_depth_texture

- Enables Shadow Mapping technique
  - Consumes lots of fillrate
- Depth-of-field effect
- Supports only NEAREST



Depth Texture

OES_depth_texture

```
// Depth texture for storing depth information
glGenTextures(1, &depthTexture);
glBindTexture(GL_TEXTURE_2D, depthTexture);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);

glFramebufferTexture2D(..., GL_DEPTH_ATTACHMENT, ..., depthTexture, 0);
```

Demo

Stencil Wrap

OES_stencil_wrap

- Performance improvement for Shadow Volume technique
- Wraps when the ray goes in and out of shadow volume

Stencil Wrap

OES_stencil_wrap

- Performance improvement for Shadow Volume technique
- Wraps when the ray goes in and out of shadow volume
- Shaders session has more details on Stencil Shadow Volumes technique

OpenGL ES Shading and Advanced Rendering	Presidio Wednesday 3:15PM

Float Textures

OES_texture_float and OES_texture_half_float

- Add new texture formats: 16-bit and 32-bit floats
  - PowerVR SGX based devices only
- Processing HDR images
- GPGPU

Float Textures

OES_texture_float and OES_texture_half_float

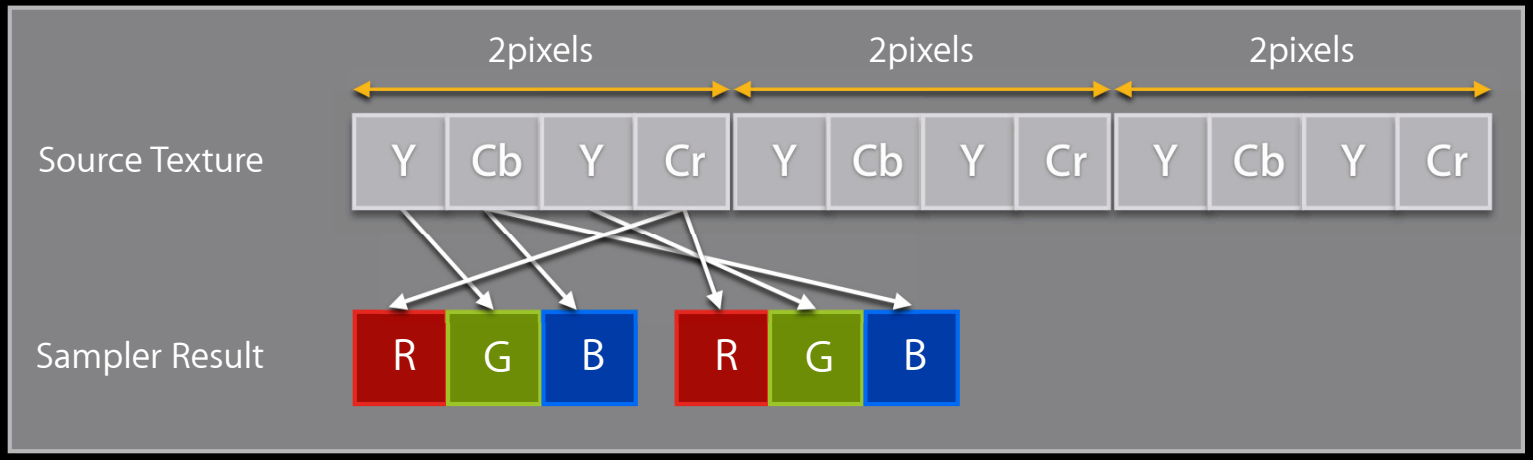
- Add new texture formats: 16-bit and 32-bit floats
  - PowerVR SGX based devices only
- Processing HDR images
- GPGPU

```
glTexImage2D(..., GL_HALF_FLOAT_OES, data);
```

Video Playback Using OpenGL

APPLE_rgb_422

- Interleaved 422 video textures
- YCbCr422 texture is sampled as RGB
- The shader must implement the color space conversion
- PowerVR SGX based devices only



Video Playback Using OpenGL

APPLE_rgb_422

```
glTexImage2D(...);
```

```
uniform sampler2D texSampler;
varying lowp vec4 colorVarying;
varying mediump vec2 interpTexcoord;

void main()
{
    lowp vec4 color;
    color = texture2D(texSampler, interpTexcoord);

      gl_FragColor = vec4(color, 1.0);
}
```

Video Playback Using OpenGL

APPLE_rgb_422

```
glTexImage2D(..., GL_RGB_422_APPLE, GL_UNSIGNED_SHORT_8_8_REV_APPLE, ...);
```

```
uniform sampler2D texSampler;
varying lowp vec4 colorVarying;
varying mediump vec2 interpTexcoord;

void main()
{
    lowp vec4 color;
    color = texture2D(texSampler, interpTexcoord);

    lowp vec3 convertedColor = vec3(-0.87075, 0.52975, -1.08175);
    convertedColor += 1.164 * color.ggg;                    // Y
    convertedColor += vec3(0.0, -0.391, 2.018) * color.bbb; // U
    convertedColor += vec3(1.596, -0.813, 0.0) * color.rrr; // V

    gl_FragColor = vec4(convertedColor, 1.0);
}
```

New Extensions

OpenGL ES in iOS 4

- APPLE_framebuffer_multisample
- EXT_discard_framebuffer
- OES_vertex_array_object
- APPLE_texture_max_level
- APPLE_shader_texture_lod
- OES_depth_texture
- OES_stencil_wrap
- OES_float_texture
- APPLE_rgb_422

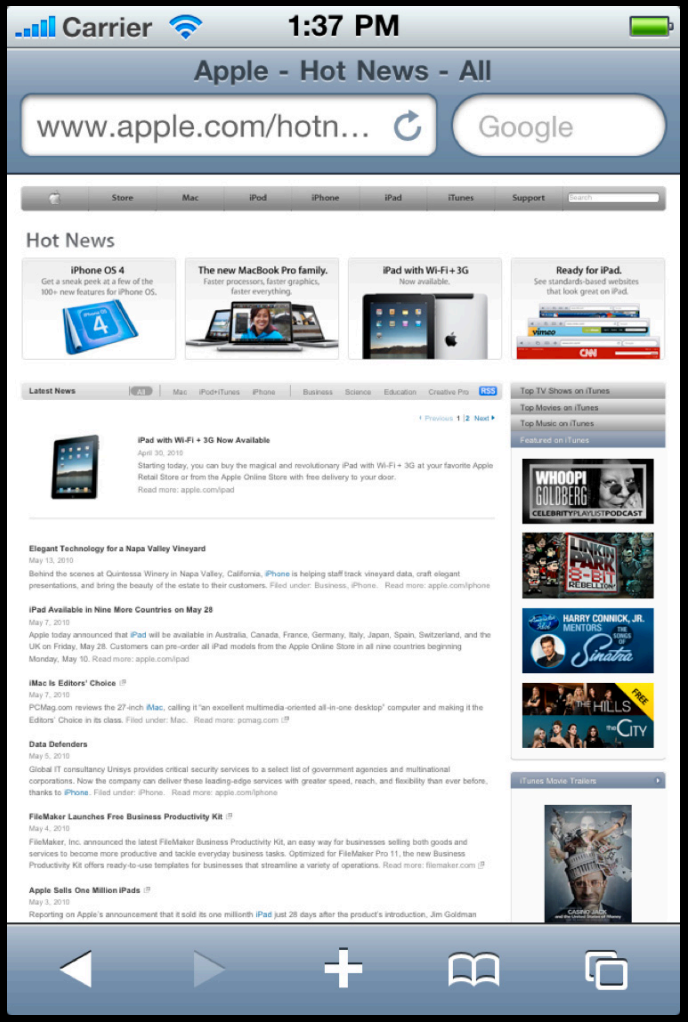
New Extensions

Retina Display

Large Display Performance

Multitasking


Carrier
1:37 PM
Apple - Hot News - All
www.apple.com/hotne...
Google
Hot News


Carrier
1:37 PM
Apple - Hot News - All
www.apple.com/hotn...
Google
Store
Mac
iPod
iPhone
iPad
iTunes
Support
Hot News
iPhone OS 4
Get a sneak peek at a few of the 100+ new features for iPhone OS.
The new MacBook Pro family.
Faster processors, faster graphics, faster everything.
iPad with Wi-Fi+3G
Now available.
Ready for iPad.
See standards-based websites that look great on iPad.
Latest News
iPad with Wi-Fi + 3G Now Available
Elegant Technology for a Napa Valley Vineyard
iPad Available in Nine More Countries on May 28
iMac Is Editors' Choice
Data Defenders
FileMaker Launches Free Business Productivity Kit
Apple Sells One Million iPads
Top TV Shows on iTunes
Top Movies on iTunes
Top Music on iTunes
Featured on iTunes
iTunes Movie Trailers


ww.apple.
Google
Store

Retina Display

Adopting High Resolution

1. Allocate high-resolution color buffer
2. Fix hard-coded sizes
3. Load higher-resolution artwork

Retina Display

Setting color buffer size

- UIView.contentScaleFactor
  - `pixelSize = view.bounds.size * view.contentScaleFactor`
  - Defaults to UIScreen.scale for most built-in UIKit views
  - Defaults to 1.0 for OpenGL views

Retina Display

Setting color buffer size

```
//  allocate color buffer
view.contentScaleFactor = [UIScreen mainScreen].scale;
[ctx renderbufferStorage: target fromDrawable: view.layer];


//  save color buffer dimensions
glGetRenderbufferParameteriv(target, RENDERBUFFER_WIDTH, &pixelWidth);
glGetRenderbufferParameteriv(target, RENDERBUFFER_HEIGHT, &pixelHeight);
```

Retina Display

Avoid hard-coded sizes

```
glScissor (GLint x, GLint y, GLsizei width, GLsizei height)
glViewport (GLint x, GLint y, GLsizei width, GLsizei height)
glReadPixels (GLint x, GLint y, GLsizei width, GLsizei height, ...)
glTexImage2D (..., GLsizei width, GLsizei height, ...)
glRenderbufferStorage (..., GLsizei width, GLsizei height)
```

- Depth buffer size must match color buffer

```
glRenderbufferStorage(target, DEPTH_COMPONENT24, pixelWidth, pixelHeight);
```

- Most applications use a full-size viewport

```
glViewport(0, 0, pixelWidth, pixelHeight);
```

Retina Display

Texture sizing

- Use higher-resolution textures
- Share assets with iPad
  - Performance and display size are similar

Retina Display

Retina Display

UIImage

- `UIImage.size` returns points, not pixels
  - `glTexImage2D` requires pixels
- Pixel dimensions = image.size × image.scale
  - or use CGImageGetWidth/Height

Retina Display

Summary

- UIKit uses points, OpenGL uses pixels
- To adopt:
  - Set `glview.contentScaleFactor = screen.scale`
  - Check Depth/Stencil renderbuffer dimensions
  - Check glViewport parameters

New Extensions

Retina Display

Large Display Performance

Multitasking

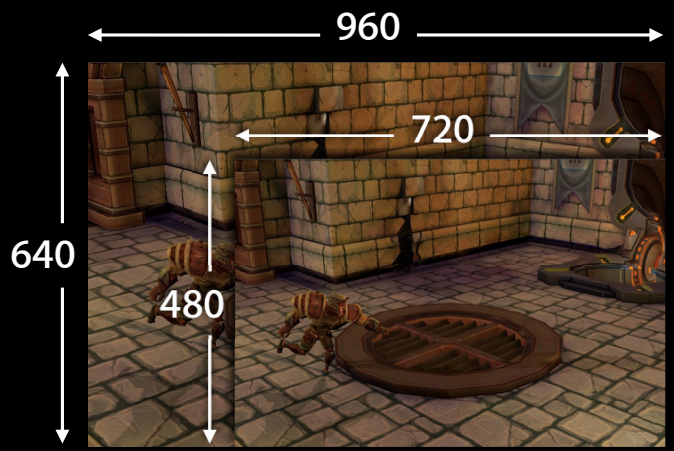
Large Display Performance

Optimizing for Large Displays

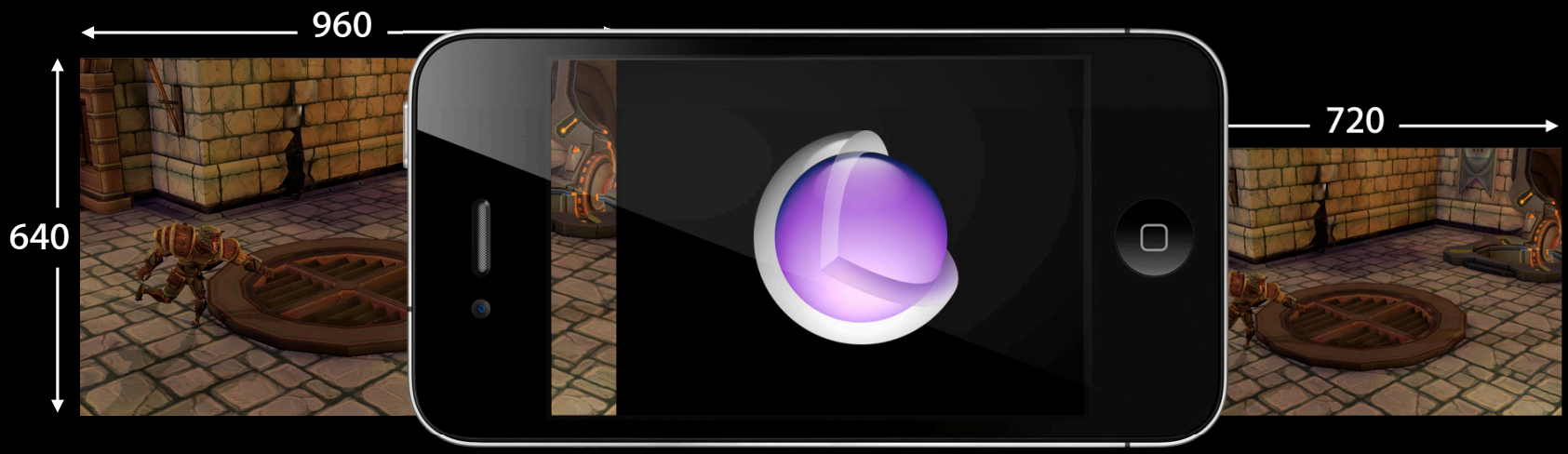
- Increased GPU workload
- How many pixels are being drawn?
- How expensive is each pixel?
  - Not using mipmaps
  - Long fragment shaders
  - Alpha Test

OpenGL ES Tuning and Optimization	Presidio Wednesday 4:30PM

Large Display Performance



Large Display Performance



Large Display Performance

When optimization isn't enough

- Render at smaller screen resolution
  - `contentScaleFactor = 1` (default)
  - `contentScaleFactor = 1`, add anti-aliasing
  - `contentScaleFactor` somewhere between 1.0 and UIScreen.scale

Performance on iPad

Renderbuffer sizing

- UIView.contentScaleFactor not available in iPhone OS 3.2
  - Use UIView.transform instead

```
CGFloat scale = ...;
CGFloat viewWidth = screenWidth * scale;
CGFloat viewHeight = screenHeight * scale;

glView.bounds = CGRectMake(0, 0, viewWidth, viewHeight);
glView.transform = CGAffineTransformMakeScale(1 / scale, 1 / scale);
```

Large Display Performance

Summary

- Investigate fill-rate optimizations
- Use smaller framebuffers to reduce fill-rate requirements
  - iOS 4: UIView.contentsScaleFactor
  - iPhone OS 3.2: UIView.transform

New Extensions

Retina Display

Large Display Performance

Multitasking


TANK
LOADING....

Multitasking

OpenGL Overview

- Fast app switching
- No GPU access in the background
  - OK to leave GPU resources allocated

Multitasking

Effective background memory usage

- Tradeoff: memory size reduction vs. time to restore
- Keep GPU resources required for a quick resume
- Delete GPU resources that are cheap to recreate
  - e.g., color and depth buffers (2-10MB)
- Delete idle GPU resources
  - e.g., cached resources not required for the current scene

Multitasking

Entering the background

- Listen for `applicationDidEnterBackground`
  - Stop GPU usage
  - Save app state
  - Release framebuffer

Multitasking

Entering the foreground

- On `applicationWillEnterForeground`
  - Allocate framebuffer
  - Begin rendering
- On `applicationDidFinishLaunching`
  - Load all required GPU resources
  - Restore saved state
  - Begin rendering

New Extensions

Retina Display

Large Display Performance

Multitasking

OpenGL ES Overview for iPhone OS

Summary

- New features to improve visual quality
  - Multisample, Float Texture, Depth Texture
- New features to improve performance
  - Vertex Array Object, Discard Framebuffer
- Retina Display adoption
- Resolution selection for large displays
- Multitasking

Related Sessions

OpenGL ES Tuning and Optimization	Presidio Wednesday 4:30PM
OpenGL ES Shading and Advanced Rendering	Presidio Wednesday 3:15PM
OpenGL Essential Design Practices	Pacific Heights Wednesday 11:30PM
Adopting Multitasking in iPhone OS, Part 1	Marina Friday 9:00AM
Optimize your iPhone App for the Retina Display	Presido Thursday 3:15PM

Labs

OpenGL for Mac OS X Lab	Graphics and Media Lab C Thursday 2:00PM
OpenGL ES Lab	Graphics and Media Lab A Thursday 9:00AM

More Information

Allan Schaffer
Graphics and Game Technologies Evangelist
aschaffer@apple.com

Documentation
OpenGL ES Programming Guide for iPhone OS
http://developer.apple.com/iphone/


WWDC10