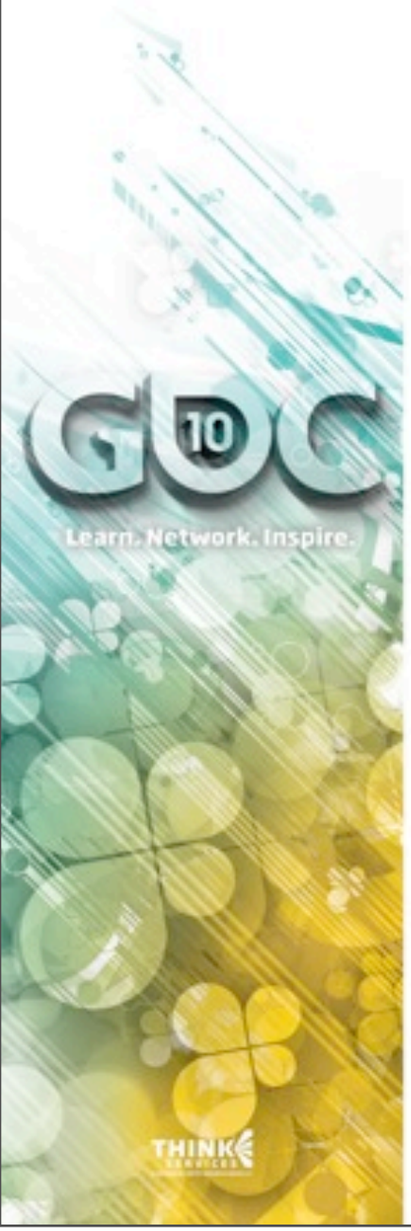
iPhoneGames SUMMIT



The Best of Both Worlds: Using UIKit with OpenGL

Noel Llopis
Snappy Touch

Twitter: @snappytouch

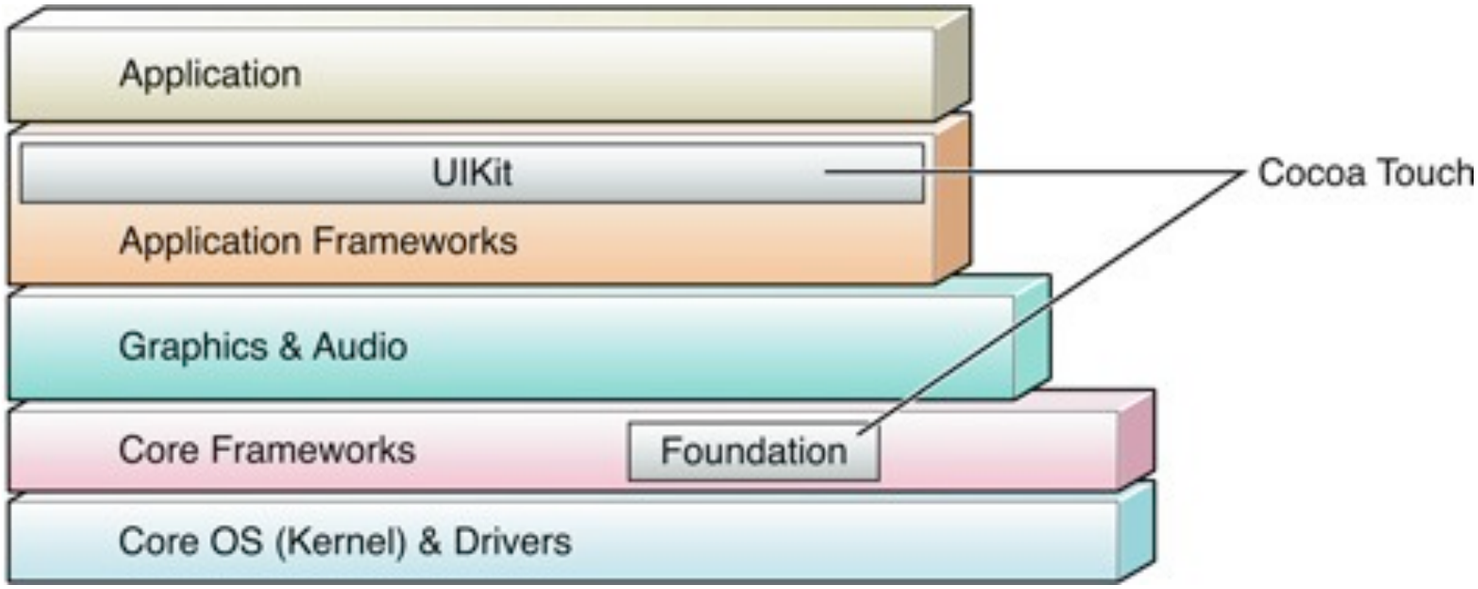
About Me

iPhone development full time for a year and a half. Flower Garden on the app store.

Many years in the games industry before that

Interesting perspective coming from console game development

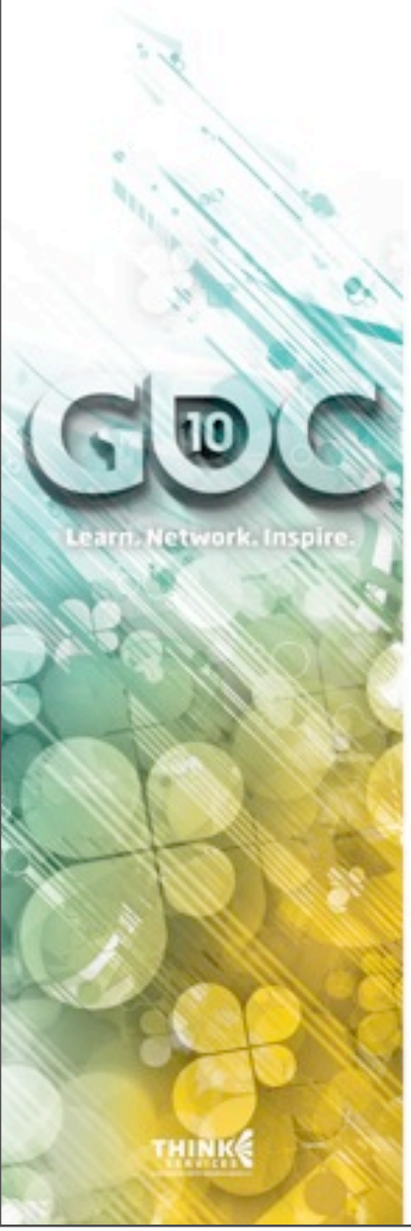
Why mix the two?


Application
UIKit
Application Frameworks
Graphics & Audio
Core Frameworks
Foundation
Core OS (Kernel) & Drivers
Cocoa Touch

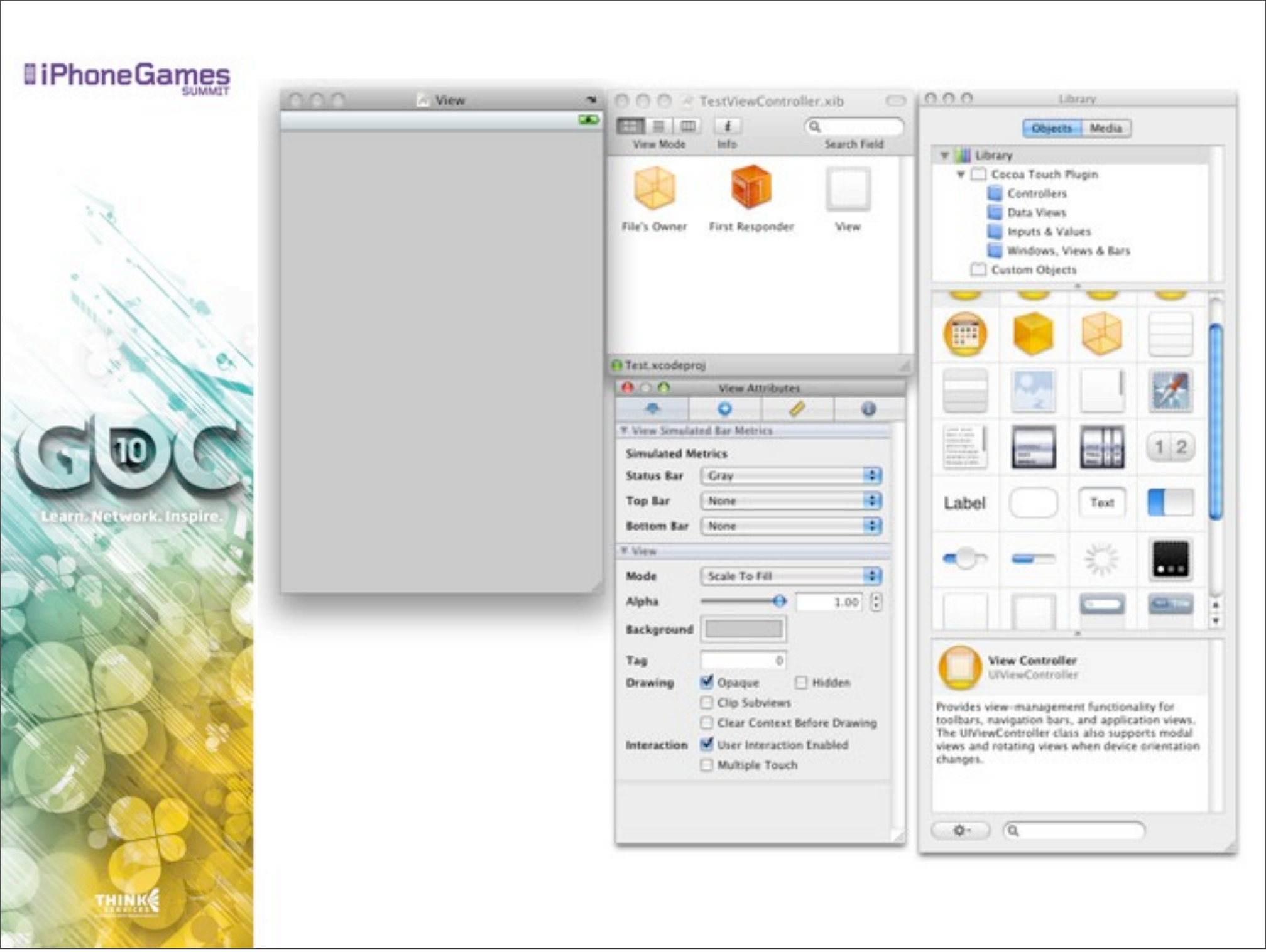
Why UIKit?

Saturday, March 13, 2010

You get all that great UI already made for you



You also get view controllers, and all the behaviors, transitions, and animations



And you also get Interface Builder!

Advantages of UIKit

- Lots of saved time!
- Familiar interface behavior
- Good performance (hardware accelerated)

Each view has its graphics uploaded to video memory (not redrawn every time)

Reasons NOT To Use UIKit

- Objective C
- Not portable (beyond iPhone/iPod Touch/iPad/MacOS)
- Not as much control over memory and performance.

- Don't be scare of objective C though. It's a great language
- Those are some of the reasons more casual games use it

1. **ALL-IN-1 GAMEBOX**
Triniti Interactive Limited
Updated Feb 15, 2010
$0.99 BUY APP

2. **Alpine Crawler World**
3D Magic Kft.
Released Mar 01, 2010
$0.99 BUY APP

3. **Doodle Jump - BE WARNE...**
Lima Sky
Updated Mar 05, 2010
$0.99 BUY APP

4. **Ragdoll Blaster 2**
Backflip Studios
Released Mar 01, 2010
$2.99 BUY APP

5. **Skee-Ball**
Freeverse, Inc.
Updated Oct 14, 2009
$0.99 BUY APP

6. **More Cupcakes!**
Maverick Software
Released Feb 23, 2010
$0.99 BUY APP

7. **Angry Birds**
Clickgamer.com
Updated Feb 12, 2010
$0.99 BUY APP

8. **Plants vs. Zombies**
PopCap Games, Inc.
Released Feb 15, 2010
$2.99 BUY APP

9. **Bejeweled® 2**
PopCap Games, Inc.
Updated Feb 11, 2010
$2.99 BUY APP

10. **Moto X Mayhem**
Occamy Games
Updated Feb 08, 2010
$0.99 BUY APP

11. **Words With Friends**
Newtoy Inc.
Updated Feb 09, 2010
$2.99 BUY APP

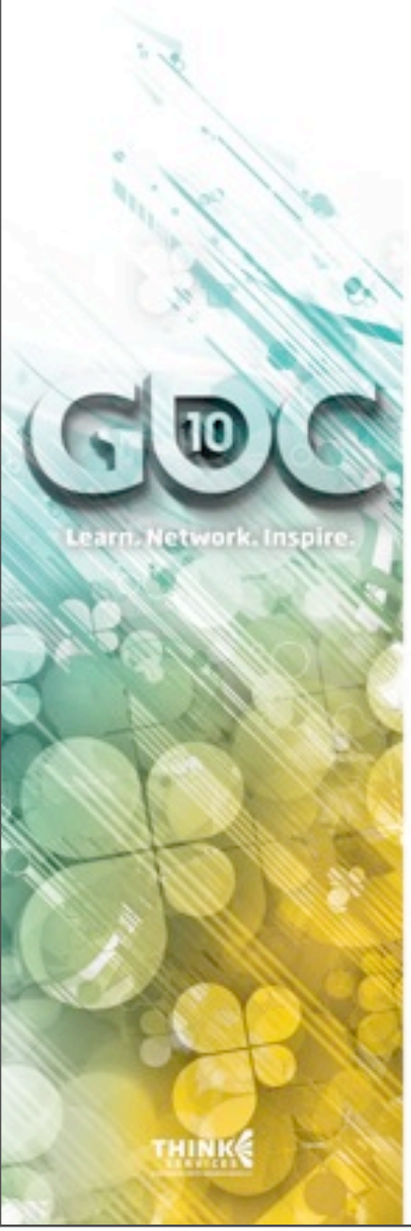
12. **MONOPOLY**
Electronic Arts
Released Nov 20, 2009
$4.99 BUY APP

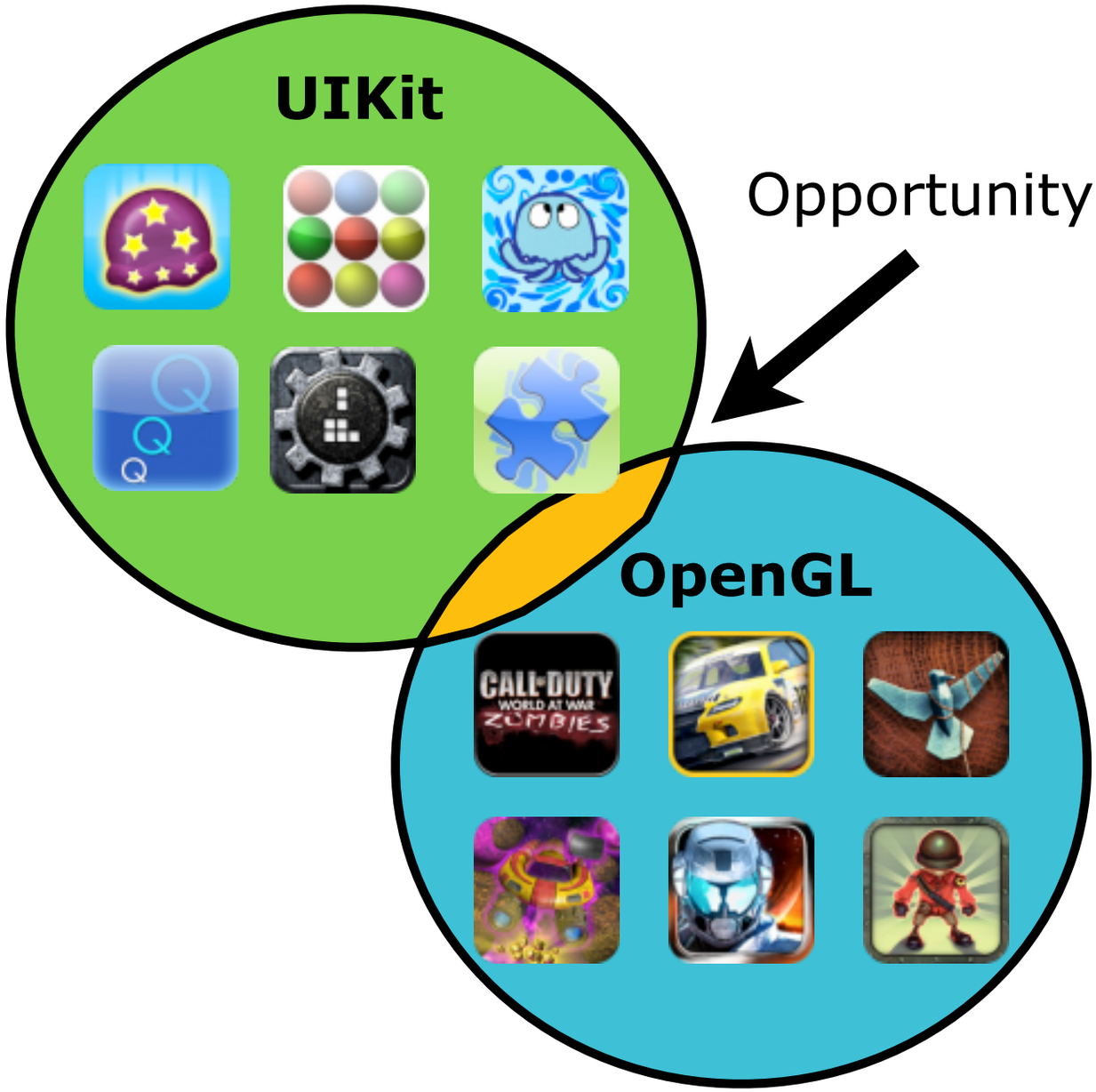
13. **Fast & Furious The Game**
I-play
Updated Aug 03, 2009
$0.99 BUY APP

14. **Doodle Army**
Chad Towns
Updated Feb 11, 2010
$0.99 BUY APP

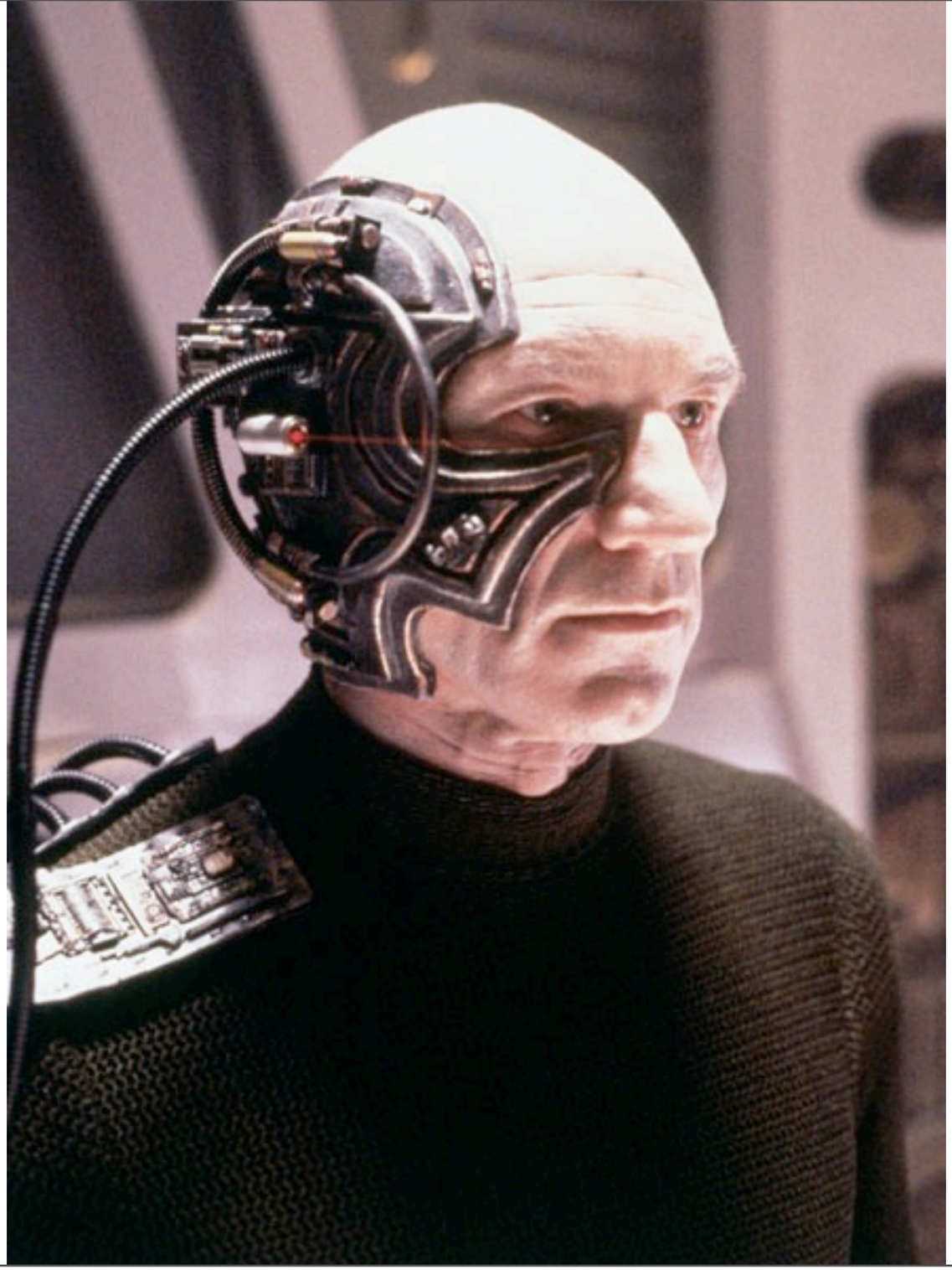
15. **FINAL FANTASY**
SQUARE ENIX Co., LTD.
Released Feb 25, 2010
$8.99 BUY APP

Are games using both today?





Usually two categories

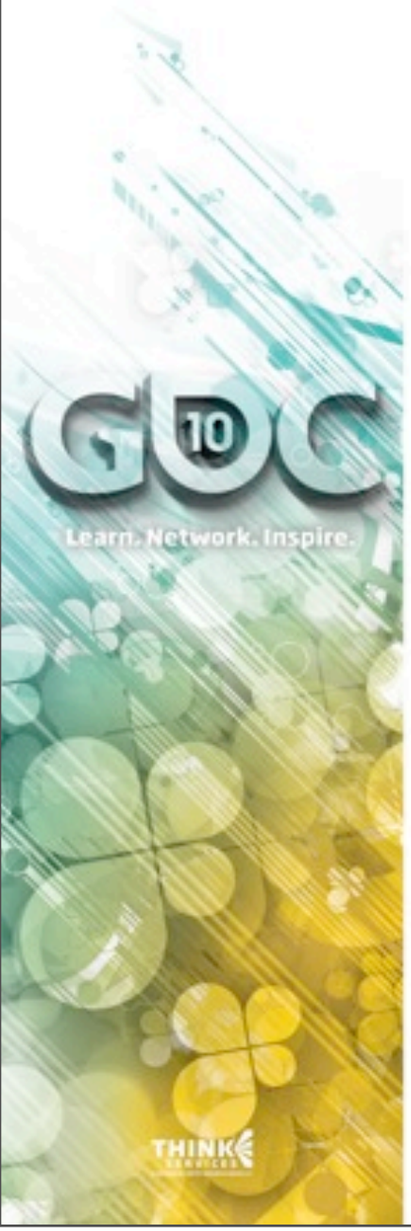


You can truly get the best of both worlds

What We're Going To See

- 0: OpenGL view
- 1: Non-fullscreen
- 2: UIKit elements
- 3: Animations
- 4: Multiple OpenGL views
- 5: Landscape orientation
- 6: Content OpenGL -> UIKit
- 7: Content UIKit -> OpenGL

iPhoneGames SUMMIT



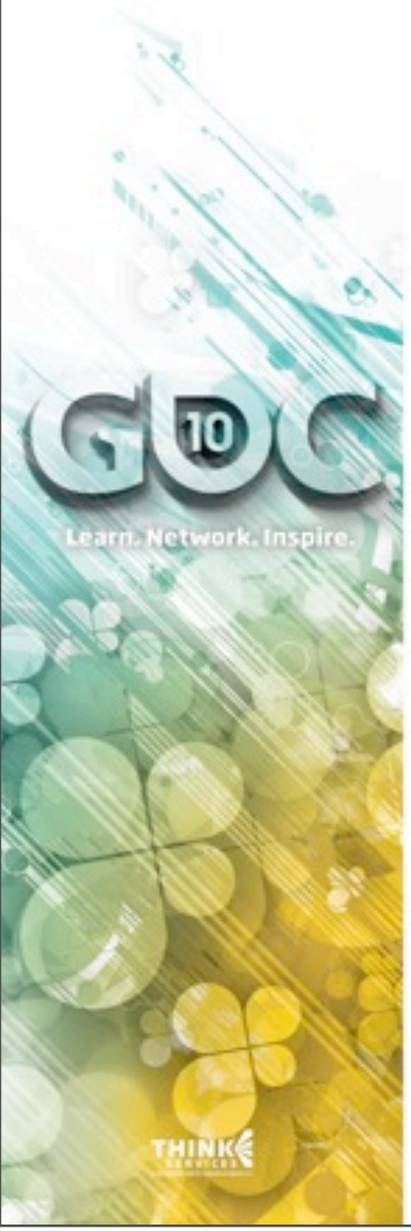
The Basics: Displaying OpenGL Graphics

The GL gravity sample from the dev site


```
@interface GLGravityView : UIView
{
  EAGLContext* context;
  ...
}

+ (Class) layerClass
{
  return [CAEAGLLayer class];
}

glBindRenderbufferOES(GL_RENDERBUFFER_OES,
                      bufferHandle);
[m_eaglContext renderbufferStorage:
          GL_RENDERBUFFER_OES
               fromDrawable:drawable];
```


That's what does the magic and allows the rendering of OpenGL to be displayed in a view.

But the point is that it's just a view, so you can do most things you can do with a regular UIView. And that's where the fun begins.



Case 1:
Not Fullscreen

Saturday, March 13, 2010

As gamers we're used to games taking up the whole screen
And that's what most games do on the iPhone as well
But it doesn't have to be that way

iPhoneGames SUMMIT

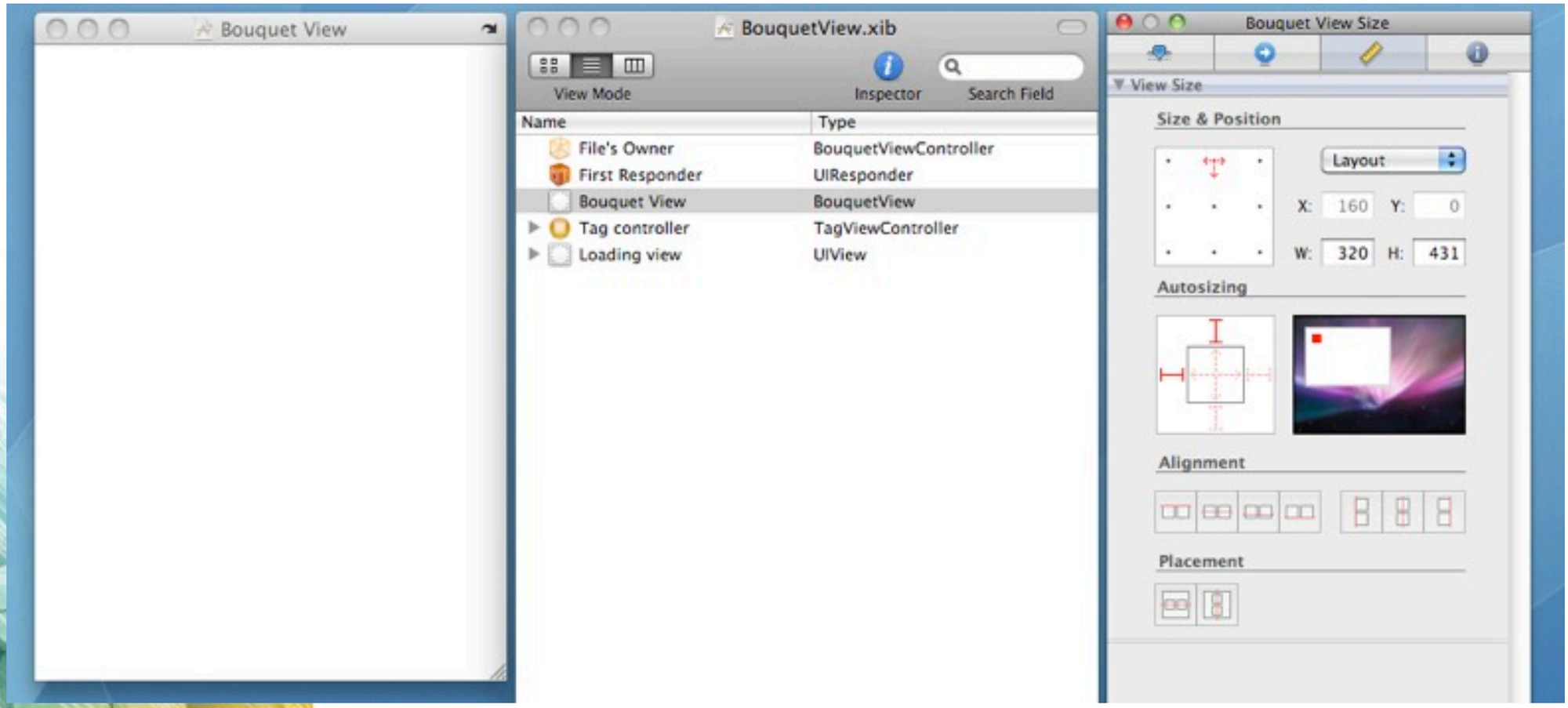
GDC 10

Learn. Network. Inspire.

THINK

OpenGL

UITabBar

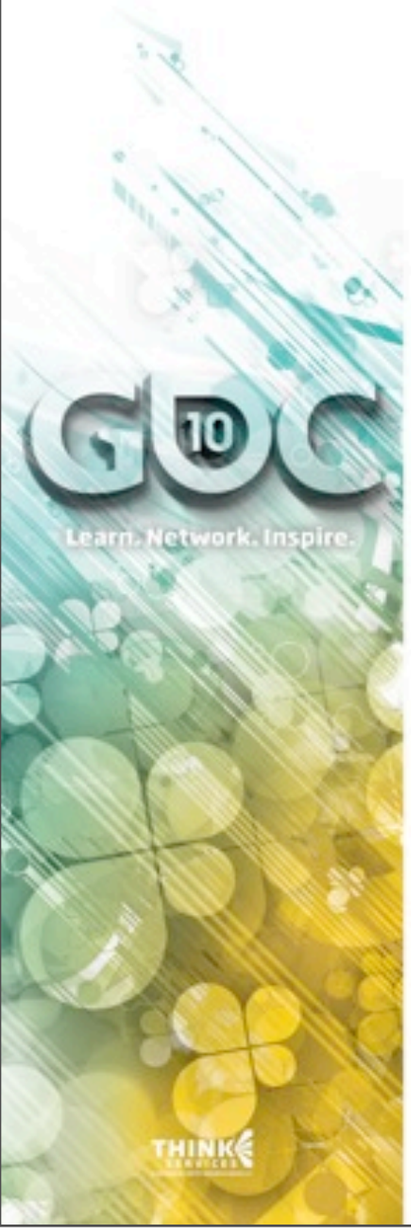

Bouquet View
BouquetView.xib
View Mode
Inspector
Search Field
Name
Type
File's Owner
BouquetViewController
First Responder
UIResponder
Bouquet View
BouquetView
Tag controller
TagViewController
Loading view
UIView
Bouquet View Size
View Size
Size & Position
Layout
X: 160
Y: 0
W: 320
H: 431
Autosizing
Alignment
Placement

320 x 431

Set correct projection matrix

Set correct viewport

Perfect performance

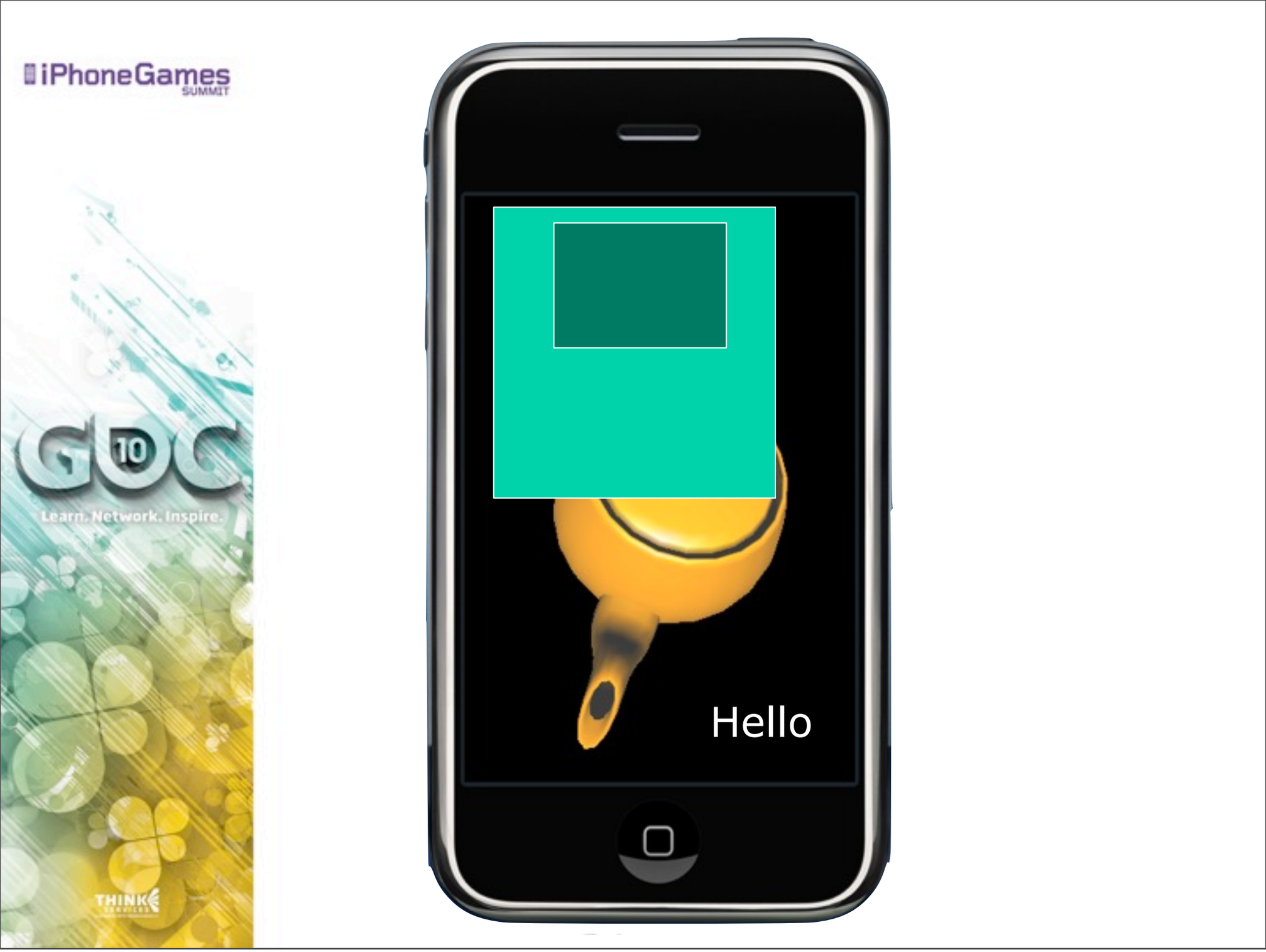


Case 2: Adding UIKit Elements On Top

Adding Subviews

- Can use addSubview: to add any children to OpenGL view.
- There used to be some vague warnings in the 2.x SDK docs about not doing that for “performance and instability” issues.

I never saw any instability



Saturday, March 13, 2010

addSubview anywhere

The problem is that the UIKit is designed to be mostly static with animations in responses to events.

Scene needs to be composed every time

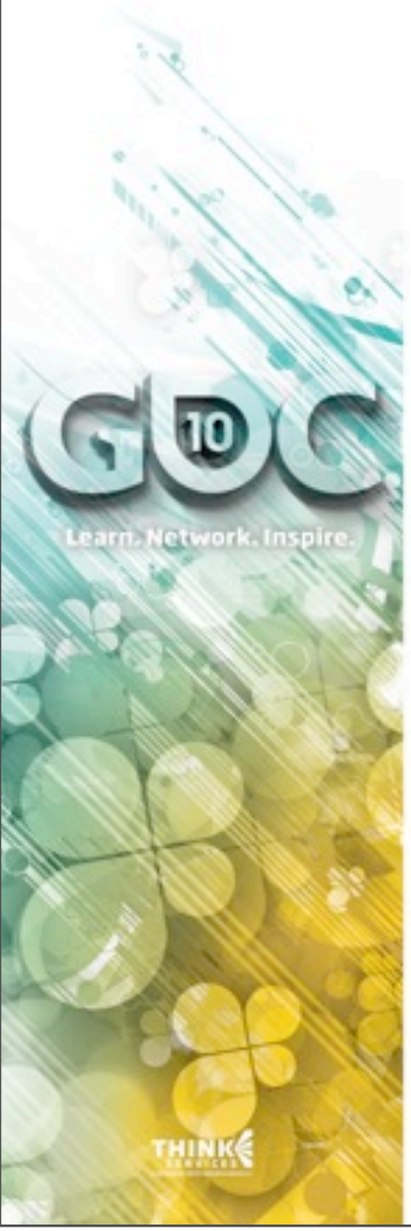
Performance Issues

- Things were particularly bad when driving main loop with NSTimer (don't do it!)
- Using CADisplayLink (3.1 or higher) seems to help a lot.
- If you display a very complex set of UIViews on top, disable update and rendering of OpenGL.

Maybe because it can coordinate better the refresh of the screen with UIKit?

Recommendations

- Avoid really complex hierarchies on top of OpenGL
- Avoid large, transparent UIKit objects
- Avoid objects that change very frequently (every frame)
- Perfect for buttons, labels, solid views



Case 3: Animating an OpenGL view

Animations

- Do it like any other view! :-)

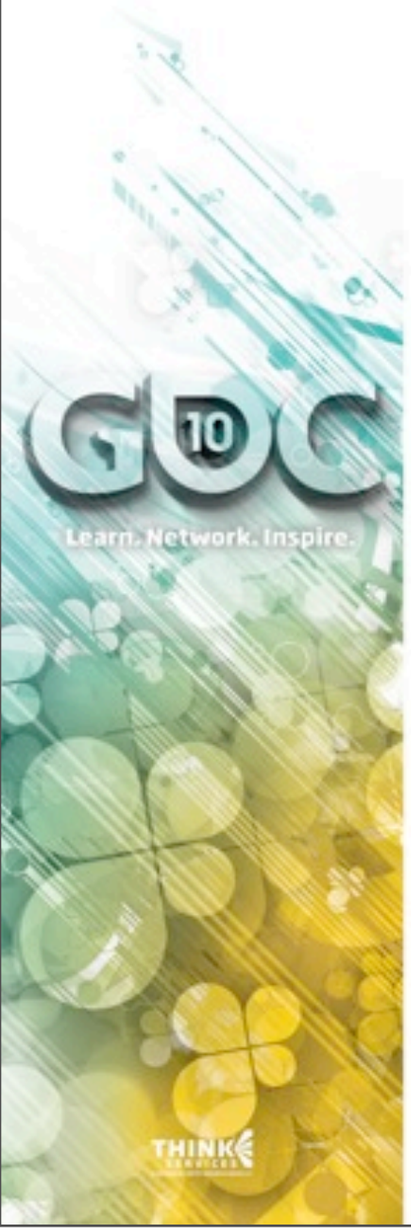
```
[UIView beginAnimations:nil context:NULL];
[UIView setAnimationDuration:0.3];
[UIView setAnimationDelegate:self];
[UIView setAnimationDidStopSelector:@selector
(tabControllerDidDisappear)];
  oldView.center = pt;
  [m_plantCareViewController view].center =
careCenter;
[UIView commitAnimations];
```


This one's easy!

Animations

- Animating a full OpenGL view seems to add a significant performance hit.
- If you can, disable update and render of OpenGL view until animation is complete.

So you can still animate it transitioning to the game, and then you can start animating the game.



Case 4: Multiple OpenGL Views

This is where it gets interesting

Why Multiple OpenGL Views?

- A lot of the time you can reuse a single OpenGL view. Especially if you're just doing full screen.
- But sometimes you need more than one: flower and bouquet screens, or main game and character customization screens.
- Sometimes you may even need to show them at the same time (transition or in same screen)

You can lay the views out in IB for convenience

Multiple OpenGL Views

- Multiple ways:
- One OpenGL context per view (prevents sharing of resources)
- One render target per view (that's what I did)

Multiple Render Targets

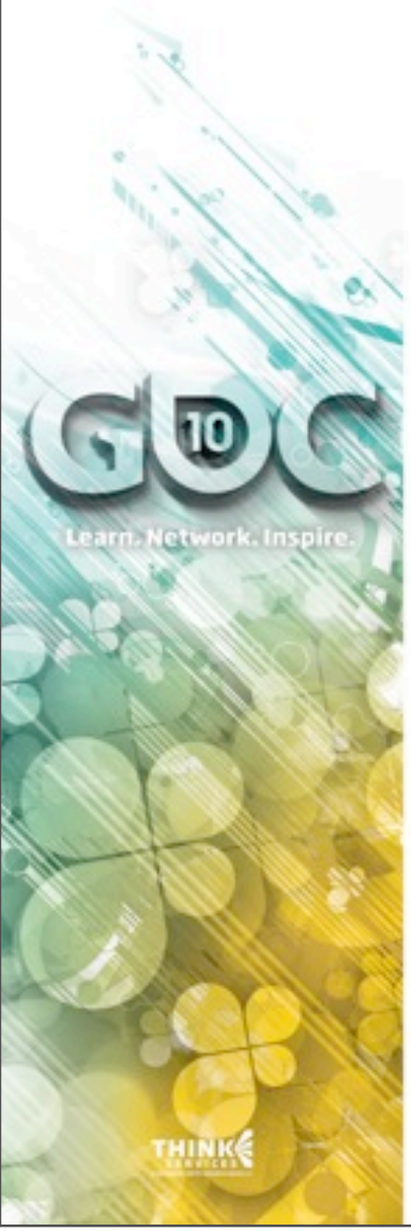
- Multiple render targets is extremely useful to render images offscreen
- Associate each OpenGL view with a new render target using that view as storage.

```
glBindFramebufferOES(GL_FRAMEBUFFER_OES,
buffer.m_frameBufferHandle);
glBindRenderbufferOES(GL_RENDERBUFFER_OES,
buffer.m_colorBufferHandle);
SetViewport(Rect(0, buffer.m_height, 0,
buffer.m_width));
```


For many uses: env maps, screenshots, advanced processing, etc

Multiple Render Targets

- Switch to each target as you render each view
- Or on viewWillAppear: if you're only switching between them.

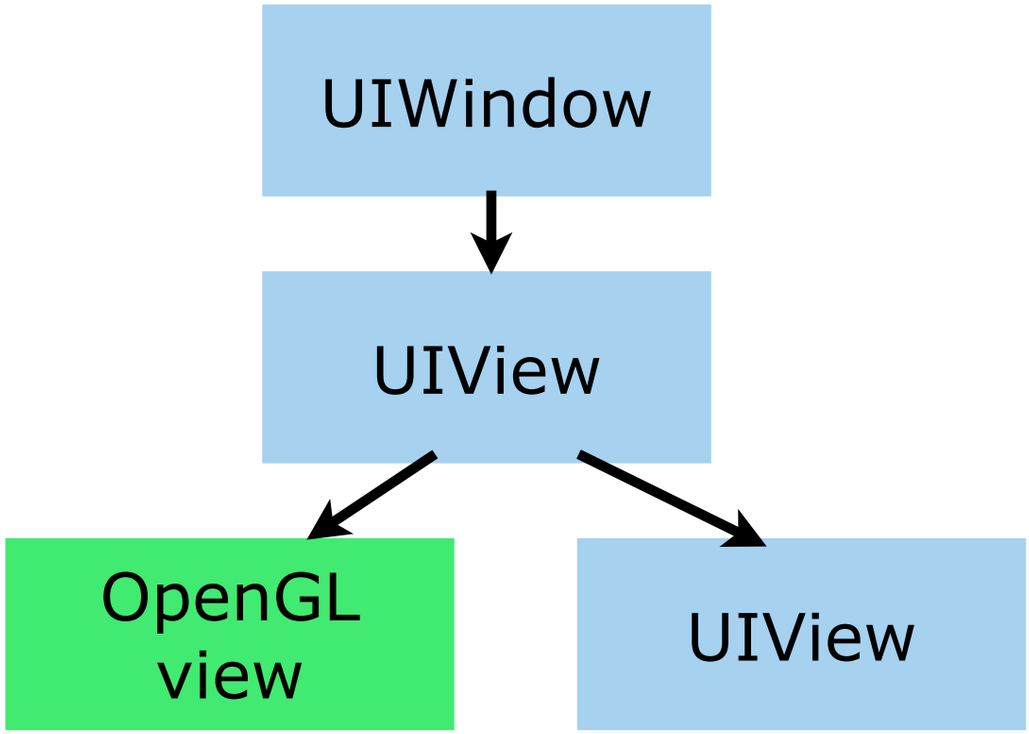


Case 5: Landscape Orientation

This one can be a bit tricky

Landscape

- You could treat the OpenGL view like any other and rotate it...



Most games are in landscape!

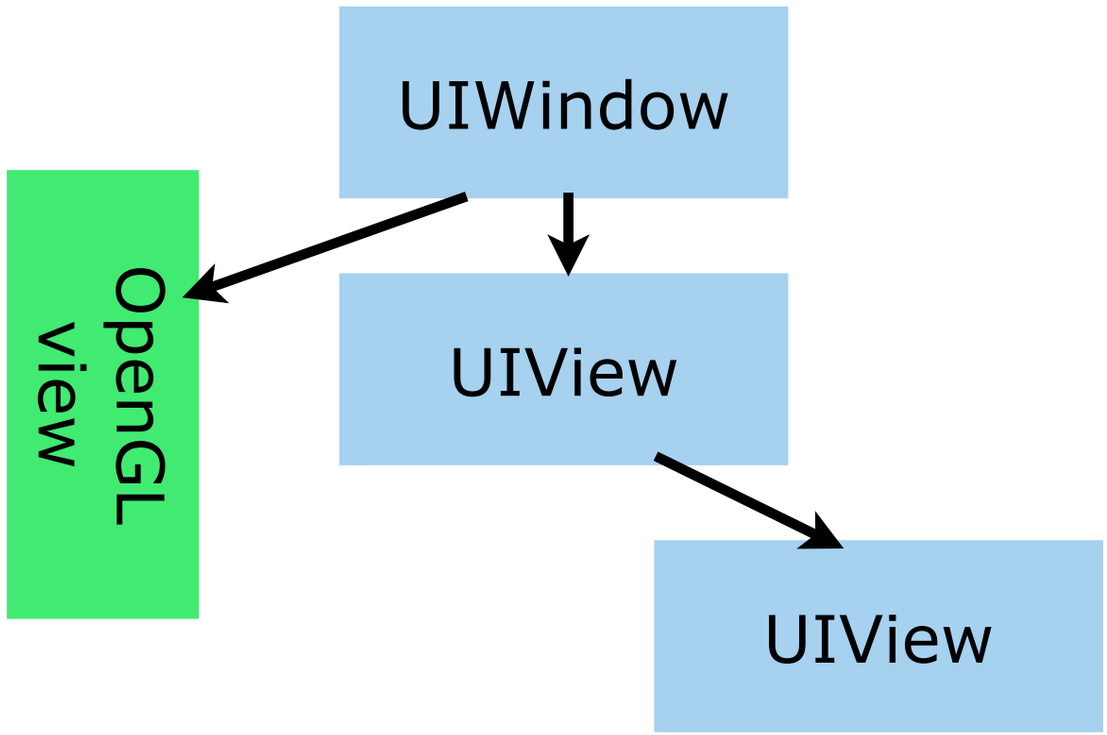
Landscape

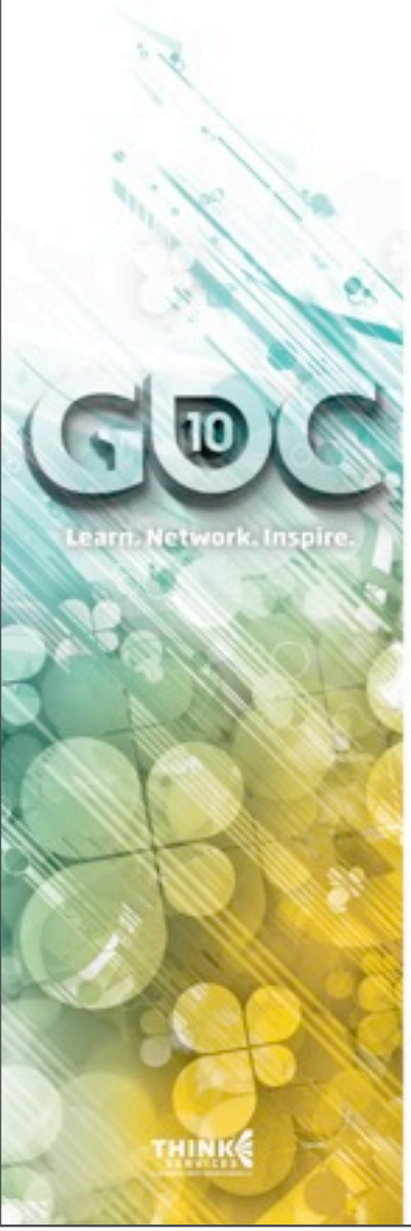
- But Apple recommends against it (for performance reasons).
- Instead, create the OpenGL view in landscape mode and set rotate your projection matrix.

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glRotatef(-90, 0, 0, 1);
glOrthof(0, 480, 0, 320, 0, 1);
```

Landscape and Hierarchy

- Since you'll use other views, you'll want to leave those rotated as usual.
- And put OpenGL view at the root.



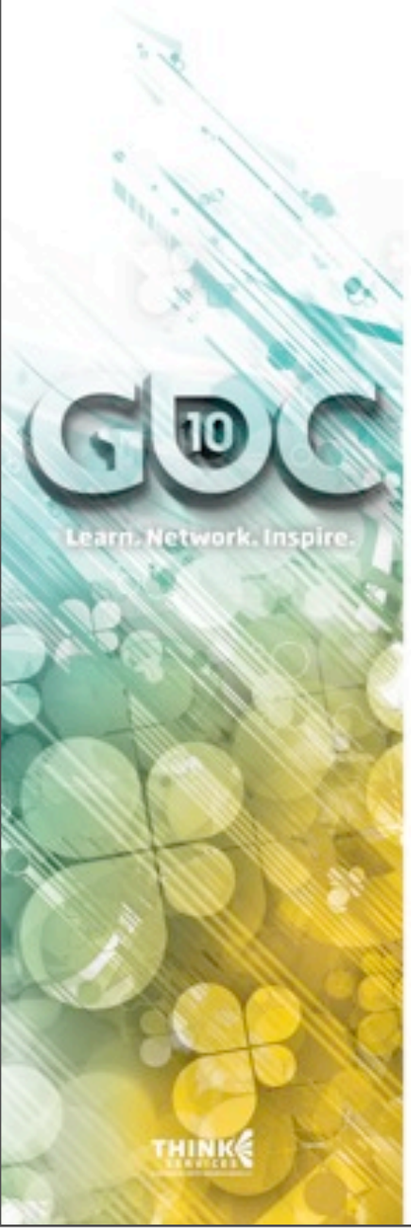


Case 6: Rendering From OpenGL To UIKit

OpenGL -> UIKit

- Whenever you want to use something you rendered in OpenGL
- For example, to save a screenshot to disk.
- Or to update an image element on a button or UIView

OpenGL -> UIKit



Flower Garden does it in two places

OpenGL -> UIKit

- The easy part is getting the pixels back: glReadPixels

```
glReadPixels(0,0,RenderTargetWidth, RenderTargetHeight, GL_RGBA, GL_UNSIGNED_BYTE, imageBuffer);
```

OpenGL -> UIKit

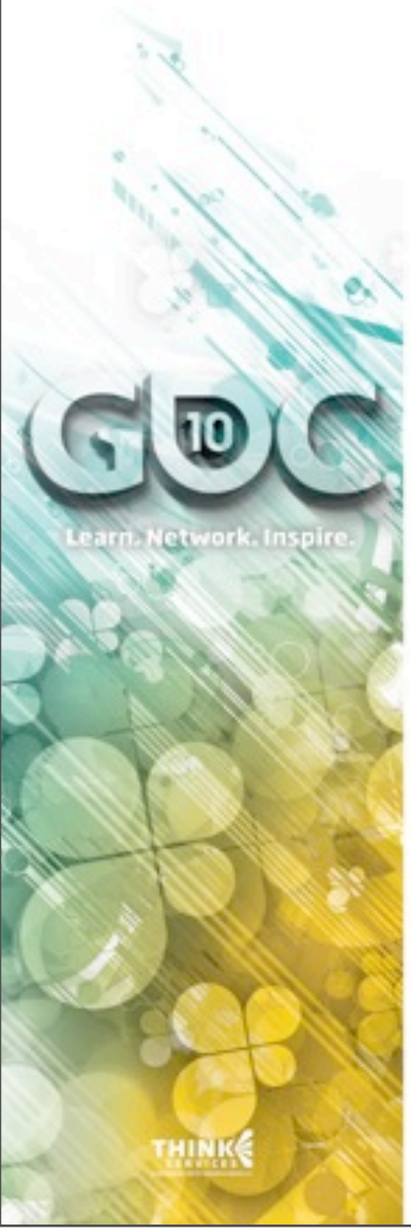
- The hard part is stuffing that into a UIImage!

```
const float RowSize = RenderTargetWidth*4;

CGDataProviderRef ref = CGDataProviderCreateWithData(NULL, imageBuffer, RenderTargetSize, NULL);
CGImageRef iref = CGImageCreate(RenderTargetWidth, RenderTargetHeight, 8, 32, RowSize,
                                        CGColorSpaceCreateDeviceRGB(),
                                        kCGImageAlphaLast | kCGBitmapByteOrderDefault, ref,
                                        NULL, true, kCGRenderingIntentDefault);

uint8_t* contextBuffer = (uint8_t*)m_resources->m_scratch.Allocate(RenderTargetSize);
memset(contextBuffer, 0, RenderTargetSize);
CGContextRef context = CGBitmapContextCreate(contextBuffer, RenderTargetWidth, RenderTargetHeight, 8, Row
                                                CGImageGetColorSpace(iref),
                                                kCGImageAlphaPremultipliedFirst | kCGBitmapByteOrder32B
CGContextTranslateCTM(context, 0.0, RenderTargetHeight);
CGContextScaleCTM(context, 1.0, -1.0);
CGContextDrawImage(context, CGRectMake(0.0, 0.0, RenderTargetWidth, RenderTargetHeight), iref);
CGImageRef outputRef = CGBitmapContextCreateImage(context);

UIImage* image = [[UIImage alloc] initWithCGImage:outputRef];

CGImageRelease(outputRef);
CGContextRelease(context);
CGImageRelease(iref);
CGDataProviderRelease(ref);
```


The gist of it is: use correct color space and flip the image
Source code on my web site

OpenGL ->UIKit

- glReadPixels is slow
- You need to create 2 temp buffers with the image data (in addition to the final UIImage). That adds up to quite a bit.
- You can use this to take higher-than-normal resolution screenshots.



Case 7: Rendering From UIKit to OpenGL

UIKit -> OpenGL

- Need to do that whenever you want to create a texture with the contents you created in UIKit.
- Font rendering
- Fancy Quartz2D bitmap creation/ composition

UIKit -> OpenGL

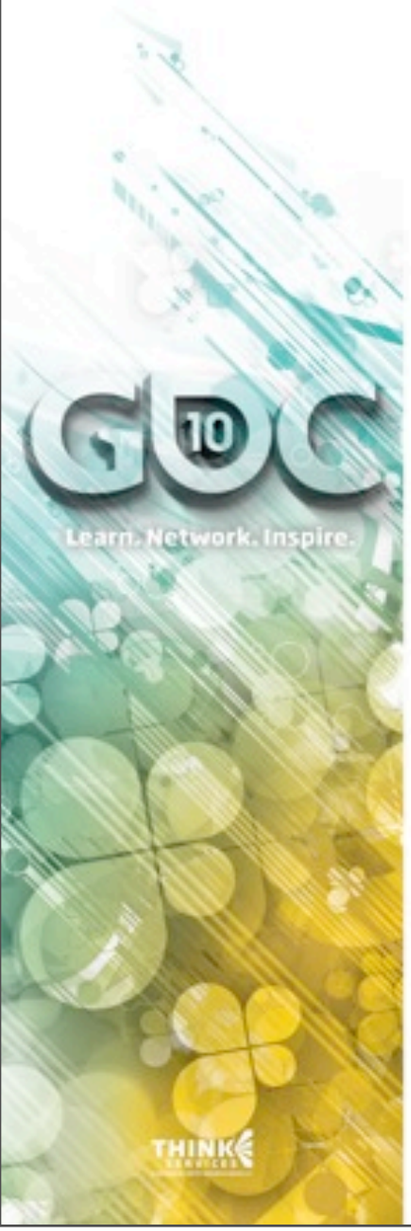
- Once you have a UIImage, do inverse conversion and set texture data.
- You can write to non 32-bit textures too.

You may be loading textures this way already (Apple samples do that)

UIKit -> OpenGL

Code to print text directly on a texture

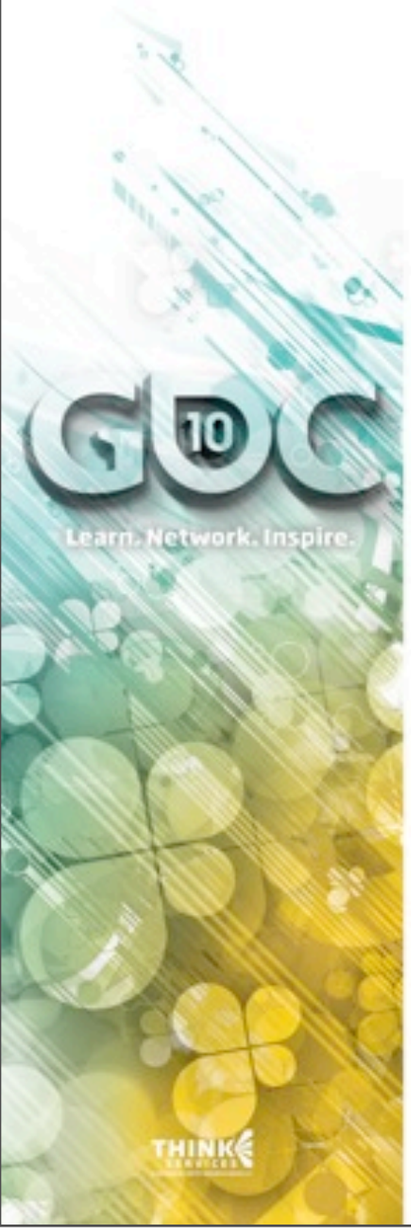
```
void TextureUtils::PrintToTexture(Texture& texture, const Rect& destRect, NSString* txt, UIFont* font, Seq
scratch)
{
    int width, height;
    texture.GetDimensions(width, height);

    CGColorSpaceRef     colorSpace = CGColorSpaceCreateDeviceGray();
    int sizeInBytes = height*width;
    void* data = scratch.Allocate(sizeInBytes);
    memset(data, 0, sizeInBytes);
    CGContextRef context = CGBitmapContextCreate(data, width, height, 8, width, colorSpace, kCGImageAlphaN
    CGColorSpaceRelease(colorSpace);
    CGContextSetGrayFillColor(context, 1.0f, 1.0f);
    CGContextTranslateCTM(context, 0.0, height);
    CGContextScaleCTM(context, 1.0, -1.0);
    UIGraphicsPushContext(context);

            [txt drawInRect:CGRectMake(destRect.left, destRect.bottom, destRect.Width(), destRect.Height()
                              withFont:font lineBreakMode:UILineBreakModeWordWrap alignment:UITextAlignmen

    UIGraphicsPopContext();

    texture.SetData(data, sizeInBytes);

    CGContextRelease(context);
    scratch.Reset();
}
```



Putting It All Together

Saturday, March 13, 2010

- OpenGL view non full screen - Multiple OpenGL views (other screens)
- UIKit elements on top - OpenGL -> UIKit (send bouquet)
- UIKit -> OpenGL text from text field to texture



Conclusions

Conclusions

- Very powerful to mix the two.
- Saves lots of time and you get access to great tools.
- Learn and appreciate Interface Builder (I turned it into a level editor for my latest project!)

Questions?

Slides and sample code on my web site

Noel Llopis
Blog: http://gamesfromwithin.com
Email: noel@snappytouch.com
Twitter: @snappytouch