# Advances in OpenGL ES for iOS 5

Session 414

**Gokhan Avkarogullari**
**Eric Sunalp**
iPhone GPU Software

# iPad 2


iPad
9:41 AM
Messages
Monday
6
Calendar
Notes
Reminders
Maps
YouTube
Videos
Contacts
Game Center
iTunes
App Store
Newsstand
FaceTime
Camera
Photo Booth
Settings
Safari
Mail
Photos
Music


1 / 1
1 / 1
105 MPH
01:21.486


1 / 1
1 / 1
105 MPH
01:21.486
Per Pixel Lighting/
Normal Maps


1 / 1
1 / 1
105 MPH
01:21.486
LightMaps
GlossMaps
SpecularMaps


1 / 1
1 / 1
86 MPH
01:06.267
BEECHERTON
Zanno
Dynamic Shadows


MSAA
1 / 1
1 / 1
86 MPH
01:06.267
BEECHERTON


Higher Poly Models

GLKit

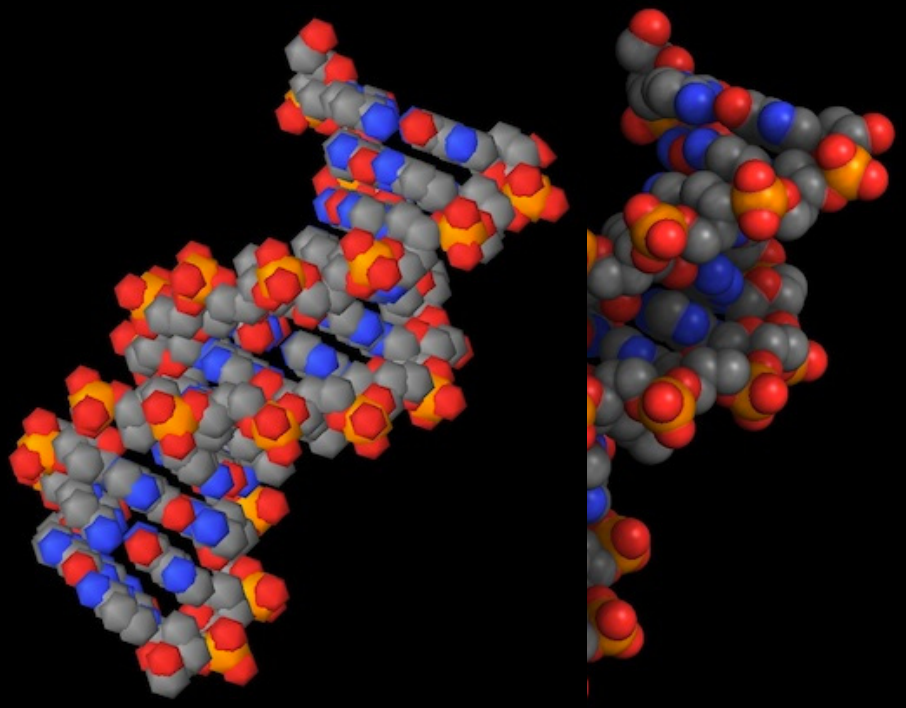
New Features

GLKit

New Features

# OpenGL ES 2.0

## Molecules.app



OpenGL ES 1.1 OpenGL ES 2.0

# GLKit

## Goals


- Making life easier for the developers
  - Find common problems
  - Make solutions available
- Encourage unique look
  - Fixed-function pipeline games look similar
    - Shaders to rescue
    - How about porting

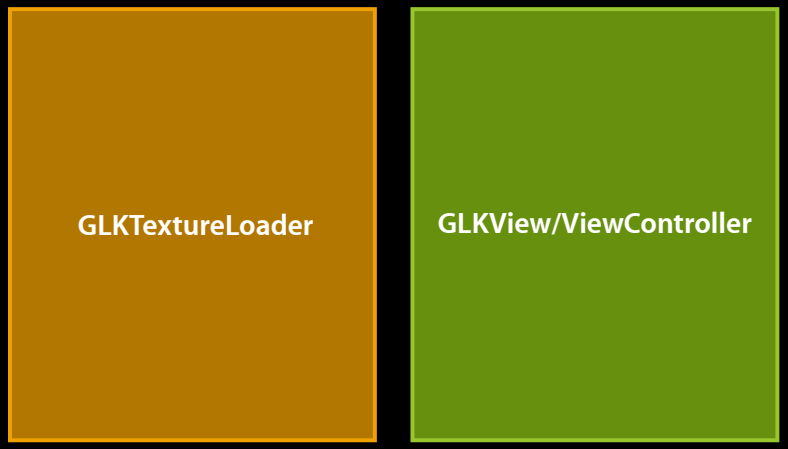
# GLKit

## GLKTextureLoader


- Give a reference—get an OpenGL Texture Object
- No need to deal ImageIO, CGImage, libjpg, libpng…

# GLKit

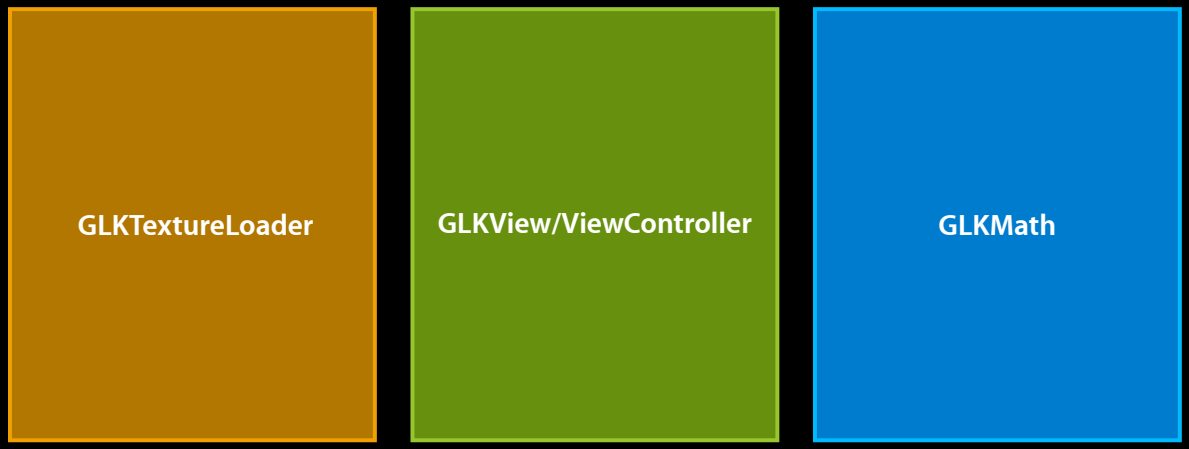
## GLKView and GLKViewController




- First-class citizen of UIKit hierarchy
- Encapsulates FBOs, display links, MSAA management…

# GLKit

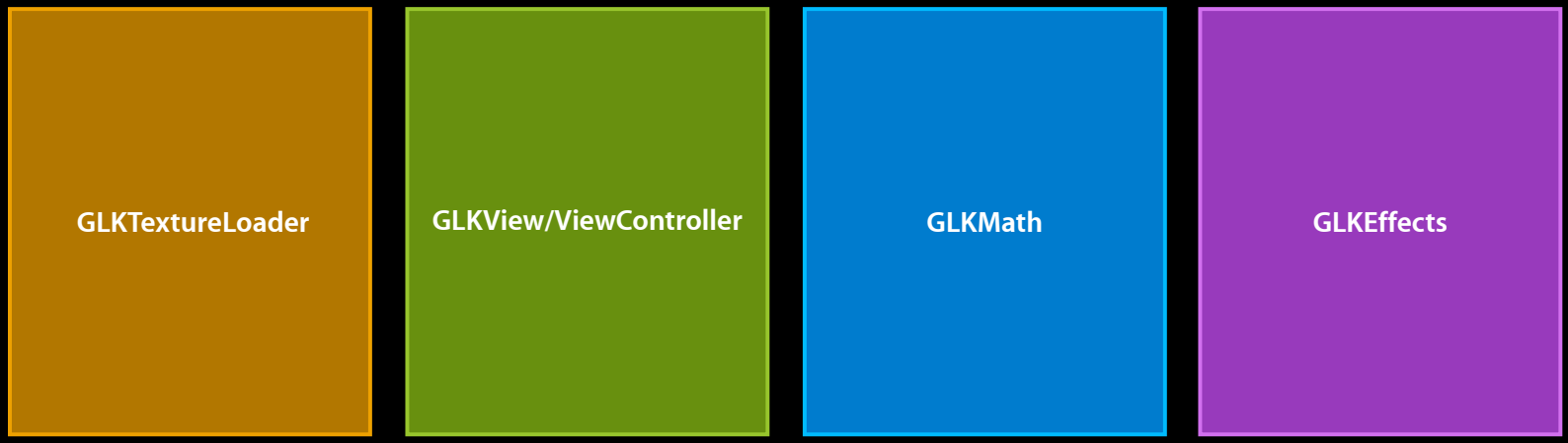
## GLKMath




- 3D Graphics math library
- Matrix stack, transforms, quaternions…

# GLKit

## GLKEffects




- Fixed-function pipeline features implemented in ES 2.0 context

# GLKTextureLoader

# GLKTextureLoader

## Overview

- Makes texture loading simple
- Supports common image formats
  - PNG, JPEG, TIFF, etc.
- Non-premultiplied data stays non-premultiplied
- Cubemap texture support
- Convenient loading options
  - Force premultiplication
  - Y-flip
  - Mipmap generation

# GLKTextureLoader

## Basic usage

- Make an EAGLContext current
- Call a GLKTextureLoader class-loading method
- Get back a `GLKTextureInfo` object
  - texture name
  - width, height
  - alpha
  - origin
  - mipmapped

# GLKTextureLoader

## Code example

```
[EAGLContext setCurrentContext:context];

NSString *path =
  [[[NSBundle mainBundle] pathForResource:@"MyImage" ofType:@"png"];

NSDictionary *options = [NSDictionary dictionaryWithObject:
  [NSNumber numberWithBool:YES] forKey:GLKTextureGenerateMipmaps];

GLKTextureInfo *textureInfo = [GLKTextureLoader
  textureWithContentsOfFile:path options:options error:error];
...

glBindTexture(GL_TEXTURE_2D, textureInfo.textureName);
...
```

# GLKTextureLoader

## Asynchronous usage

- Create a GLKTextureLoader object using the context's sharegroup
- Call an instance-loading method with a GCD queue and completion block
- Get back a GLKTextureInfo object as a parameter to the completion block

# GLKTextureLoader

## Asynchronous code example

```
NSString *path =
  [[NSBundle mainBundle] pathForResource:@"MyImage" ofType:@"png"];

EAGLSharegroup *sharegroup = [context sharegroup];

GLKTextureLoader *loader =
  [[GLKTextureLoader alloc] initWithSharegroup:sharegroup];

[loader textureWithContentsOfFile:path options:nil queue:NULL
  completionHandler:^(GLKTextureInfo *textureInfo, NSError *error)
                {
                    [self textureLoadComplete:textureInfo];
                 }];
```

# GLKView and GLKViewController

# GLKView

## Overview

- Everything needed to get OpenGL ES in a view and on-screen
  - UIView subclass
- A delegate or subclass can be used for drawing
- Automatically handles
  - Drawable creation and deletion
    - Color, Depth, Stencil, MSAA, and discard
  - Setting the context and drawable current before a draw
  - Presenting the drawable after a draw
- Snapshot support

# GLKViewController

## Overview

- UIViewController subclass
  - Fits into the existing view controller model
- Handles the redrawing of a GLKView
  - Configurable preferred frames per second
    - Determined based on the display the view resides
  - Pause and resume (manually or when backgrounded)
- Provides an update method in sync with draw
- Provides a number of statistics

# GLKView and GLKViewController

## Delegate code example

```
- (BOOL)application:(UIApplication *)application
  didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    GLKView *glkView = (GLKView *)self.viewController.view;

    glkView.delegate = game;
    glkView.context = [game context];

    glkView.drawableColorFormat = GLKViewDrawableColorFormatRGBA8888;
    glkView.drawableDepthFormat = GLKViewDrawableDepthFormat24;
    glkView.drawableMultisample = GLKViewDrawableMultisample4X;


    self.viewController.delegate = game;
    self.viewController.preferredFramesPerSecond = 30;
}
```

# GLKView and GLKViewController

## Delegate code example

```
// methods declared in the game object's class implementation

- (void)glkViewControllerUpdate:(GLKViewController *)controller
{
    // update logic here
}

- (void)glkView:(GLKView *)view drawInRect:(CGRect)rect
{
    // draw logic here
}
```

# GLKView and GLKViewController

## Subclassing

- The GLKView subclass implements

```
- (void)drawRect:(CGRect)rect;
```

- The GLKViewController subclass implements

```
- (void)update;
```

# GLKMath

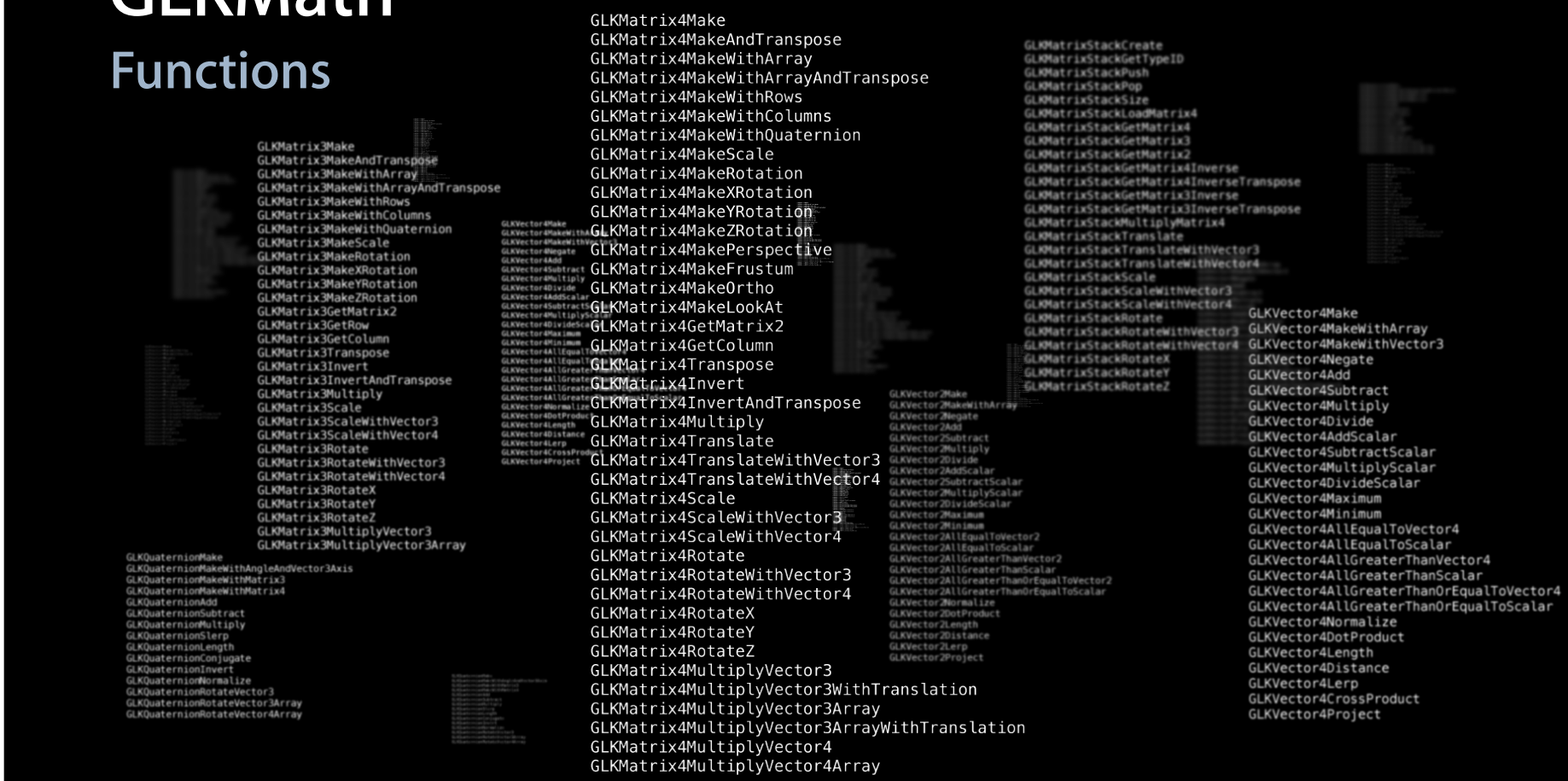
# GLKMath

## Overview

- 3D graphics math library
  - Over 175 functions
  - 4x4 and 3x3 matrix type
  - 4, 3, and 2 component vector type
  - Quaternion type
- Simplify OpenGL ES 1.1 to OpenGL ES 2.0 migration
  - Matrix stack
  - Equivalent OpenGL ES 1.1 math functions
- High performance

# GLKMath

## Functions



# GLKMath

## GLKMath example

```
GLKMatrixStack code example
GLKMatrixStackRef *projStack = GLKMatrixStackCreate(NULL);
GLKMatrixStackRef *mvStack = GLKMatrixStackCreate(NULL);

GLKMatrix4 projMat = GLKMatrix4MakeFrustum(-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
GLKMatrixStackLoadMatrix4(projection, projMat);

GLKMatrixStackLoadMatrix4(mvStack, GLKMatrix4Identity);
GLKMatrixStackTranslate(mvStack, 0.0, 0.0, -10.0);
GLKMatrixStackRotate(mvStack, M_PI, 1.0, 0.0, 0.0);
GLKMatrixStackRotate(mvStack, rotation, 0.0, 1.0, 0.0);
GLKMatrixStackScale(mvStack, scale, scale, scale);
```

# GLKEffects

# GLKEffects

## Great visual effects with minimal effort

- OpenGL ES 2.0 compatible effects library
  - Excellent segue from OpenGL ES 1.1
  - Interoperable with custom OpenGL 2.0 shaders
- 3 Effect Classes
  - GLKBaseEffect
  - GLKReflectionMapEffect
  - GLKSkyboxEffect

# GLKEffects

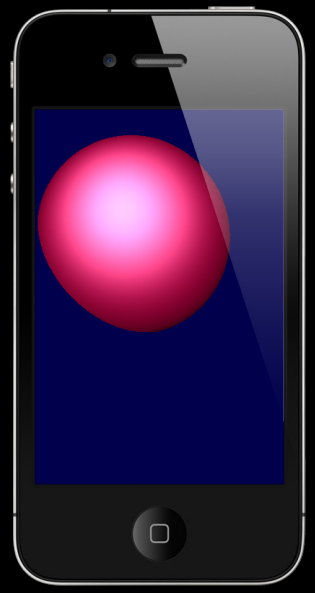
## General architecture and usage

- Configure vertex state
  - Standard OpenGL ES 2.0 vertex state setup
  - Must use predefined GLKEffects vertex attribute names
- alloc/init GLKEffect class instance
- Configure effect parameters
- Call [myEffect prepareToDraw] method
- Bind your VAO
- Call glDrawArrays() or glDrawElements()

# GLKBaseEffect

## OpenGL ES 1.1 capabilities

- Provides the most commonly used OpenGL ES 1.1 capabilities
  - Lighting
  - Materials
  - Multitexturing
  - Fog
  - Constant color
  - Transformations



# GLKBaseEffect

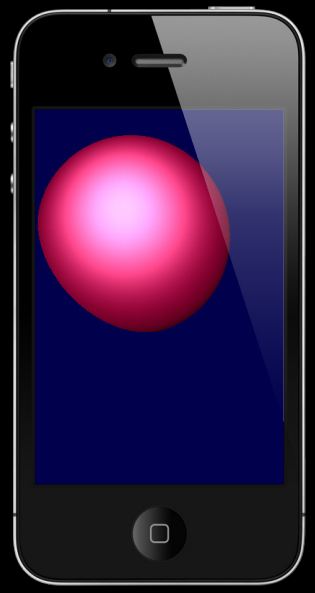
## Example—per-pixel directional lighting

```
// Initialize Vertex Array Object State (vao)
glGenVertexArraysOES(1, &vao);

glGenBuffer(1, &positionVBO);
glGenBuffers(1, &normalVBO);

glBindBuffer(GL_ARRAY_BUFFER, positionVBO);
glBufferData(GL_ARRAY_BUFFER, ...);
glVertexAttribPointer(GLKVertexAttribPosition, ...);
glEnableVertexAttribArray(GLKVertexAttribPosition);

glBindBuffer(GL_ARRAY_BUFFER, normalVBO);
glBufferData(GL_ARRAY_BUFFER, ...);
glVertexAttribPointer(GLKVertexAttribNormal, ...);
glEnableVertexAttribArray(GLKVertexAttribNormal);
```



# GLKBaseEffect

## Example—per-pixel directional lighting

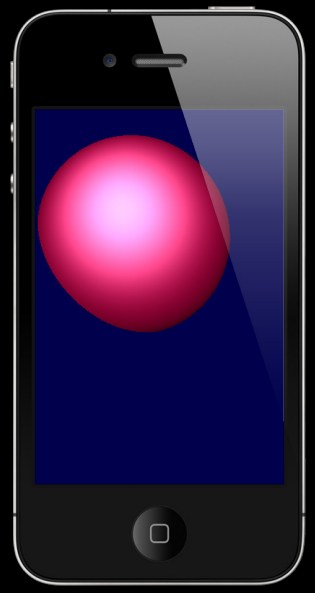
```
#import <GLKit/GLKit.h>

// Create and configure GLKBaseEffect instance
GLKBaseEffect *baseEffect = [[GLKBaseEffect alloc] init];

// Set the lighting type (per-vertex by default)
baseEffect.lightingType = GLKLightingTypePerPixel;

// Set some light and material properties
baseEffect.light0.enabled = GL_TRUE;
baseEffect.light0.diffuseColor =
      GLKVector4Make(0.8, 0.5, 0.0, 1.0);

baseEffect.material.shininess = 10.0;
```



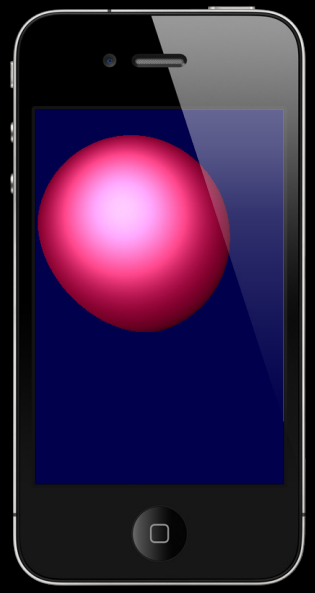
# GLKBaseEffect

## Example—per-pixel directional lighting

```
// Synchronize our effect changes
[baseEffect prepareToDraw];

// Bind our Vertex Array Object
glBindVertexArrayOES(vao);

// Draw
glDrawArrays(GL_TRIANGLE_STRIP, 0, vertCt);
```



# GLKSkyboxEffect

## A skybox in just a few steps

- GLKSkybox provides a scene skybox for your app
- Skybox has simple parameters to center and size
- No vertex array state initialization required

# GLKSkyboxEffect

## Example—skybox effect

```
// (1) Create our effect
GLKSkyboxEffect *skyboxEffect =
    [[GLKSkyboxEffect alloc] init];

// (2) Configure our cube map texture for the skybox
skyboxEffect.textureCubeMap.glName =
  [GLKTextureLoader cubeMapWithContentsOfFile:
     @"skybox.pvr" options: nil error: nil].name;

// (3) Draw the skybox
[skyboxEffect prepareToDraw];
[skyboxEffect draw];
```

# GLKReflectionMapEffect

## Cube-map reflection mapping

- GLKReflectionMap extends GLKBaseEffect
  - Reflections can be combined with the multitexturing, lighting, and fog effects of GLKBaseEffect
- Works as specified in the OpenGL 2.1 desktop specification
- Uses a cube map for reflection samples

# GLKReflectionMapEffect

## Example—reflection mapping

```
// (1) Initialize Vertex Array Object (vao)

// (2) Create our effect
GLKReflectionMapEffect *reflectionMapEffect =
    [[GLKReflectionMapEffect alloc] init];

// (3) Assign cube map texture
reflectionMapEffect.textureCubeMap.glName =
  [GLKTextureLoader cubeMapWithContentsOfFile: @"skybox.pvr",
     nil, NULL].name;
```

# GLKReflectionMapEffect

## Example—reflection mapping

```
// (4) Construct our modelview matrix
GLKMatrix4 modelMat = GLKMatrix4MakeTranslation(x, y, z);

// Create a viewing matrix per the OpenGL red book
GLKMatrix4 viewMat =
     GLKMatrix4Multiply(azimuthRot, elevRot);

// Assign our modelviewMatrix to the effect
reflectionMapEffect.transform.modelviewMatrix =
    GLKMatrix4Multiply(modelMat, viewMat);

// (5) Configure our reflection map matrix
reflectionMapEffect.matrix =
    GLKMatrix4GetMatrix3(GLKMatrix4Transpose(modelMat));
```

# GLKReflectionMapEffect

## Example—reflection mapping

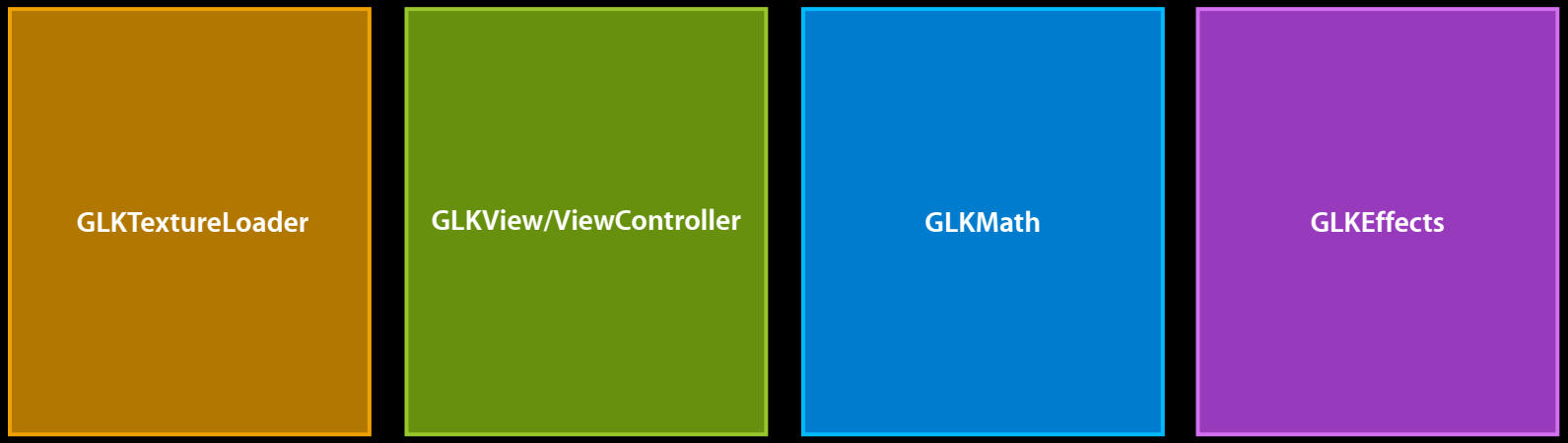
```
// (6) Get ready to draw
[reflectionMapEffect prepareToDraw];

// (7) Bind our Vertex Array Object
glBindVertexArrayOES(vao);

// (8) Draw our reflection mapped scene / object
glDrawArrays(GL_TRIANGLE_STRIP, 0, vertCt);
```

# Demo

# GLKit


GLKTextureLoader
GLKView/ViewController
GLKMath
GLKEffects

GLKit

New Features

# New Features on OpenGL ES

- Enhancing OpenGL/AVFoundation Interaction
- Enhancing the Image Quality
- Enhancing the Performance

# Enhancing OpenGL/AVFoundation Interaction

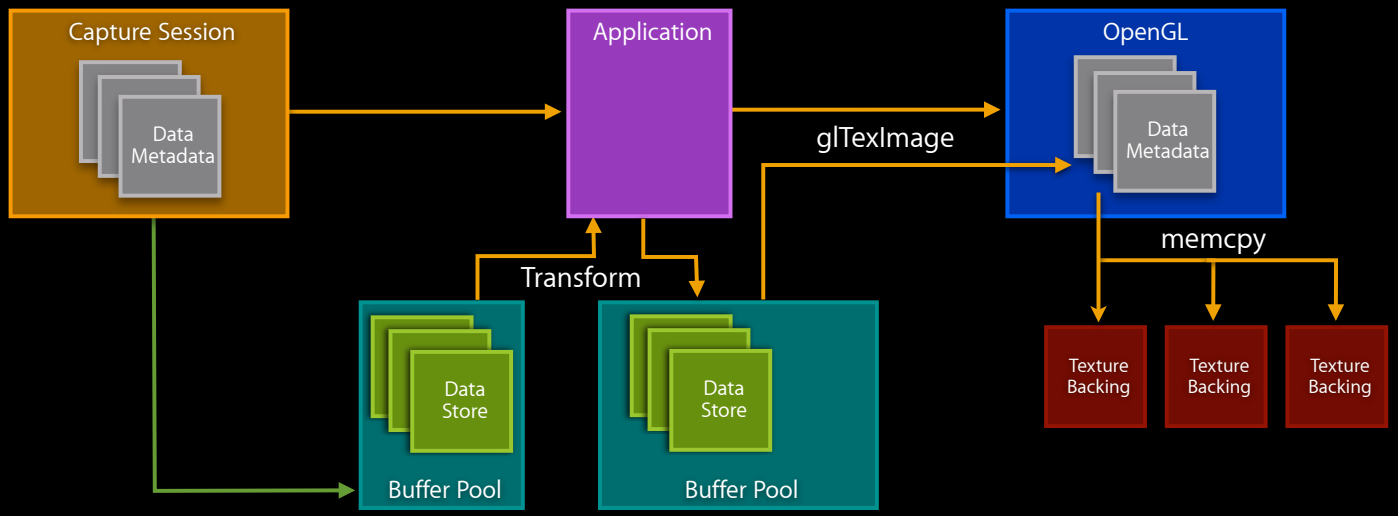
# Video and Graphics

## Purpose

- A direct path for video data to OpenGL ES
- A direct path for OpenGL ES rendering to video

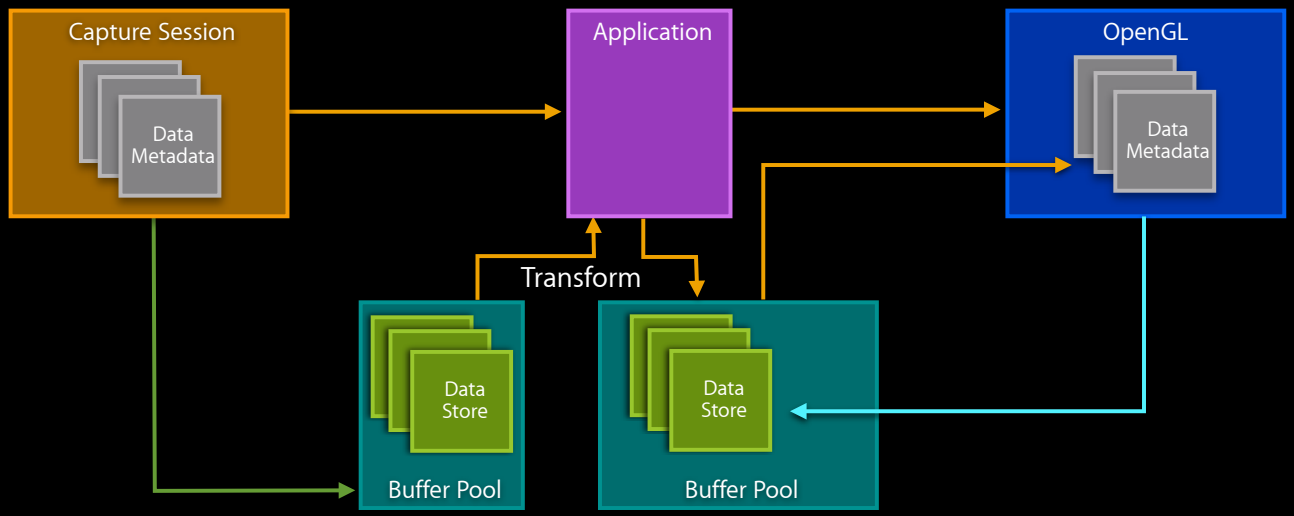
# Video and Graphics

## iOS—An indirect path for video data to OpenGL (ARGB)



# Video and Graphics

## iOS 5—A direct path for video data to OpenGL (ARGB)



# Video and Graphics

## Direct path

- iOS 5 introduces changes to avoid copies
  - OpenGL ES
    - One (R) and two component (RG) textures and render targets
  - AVFoundation
    - CVOpenGLESTextureCache

# R and RG Textures

## APPLE_texture_rg

- One component RED
  - To process luma
- Two component RED-GREEN
  - To process interleaved chroma
- Can be used both for texturing and rendering
- iPad 2 only

# CVOpenGLESTextureCache

## Usage details

```
CVReturn err = CVOpenGLESTextureCacheCreate(...&videoTextureCache);


CVImageBufferRef pixelBuffer = CMSampleBufferGetImageBuffer(sampleBuffer);


CVOpenGLESTextureRef texture = NULL;
CVReturn err = CVOpenGLESTextureCacheCreateTextureFromImage(...,
     videoTextureCache, pixelBuffer, ..., &texture);


glBindTexture(CVOpenGLESTextureGetTarget(texture),
     CVOpenGLESTextureGetName(texture));
glTexParameteri(...);

CFRelease(texture);
```

# Demo

# Enhancing the Image Quality

# High Dynamic Range Rendering

# High Dynamic Range Rendering

# High Dynamic Range Rendering

## Goal

- Within the same frame to capture wider brightness range
- Display is still 24-bit
  - Render to a texture that is high dynamic range
  - Tone map
- HDR rendering enables
  - Brightness changes
    - Outdoors vs. indoors
  - Blooms
  - Better motion blurs

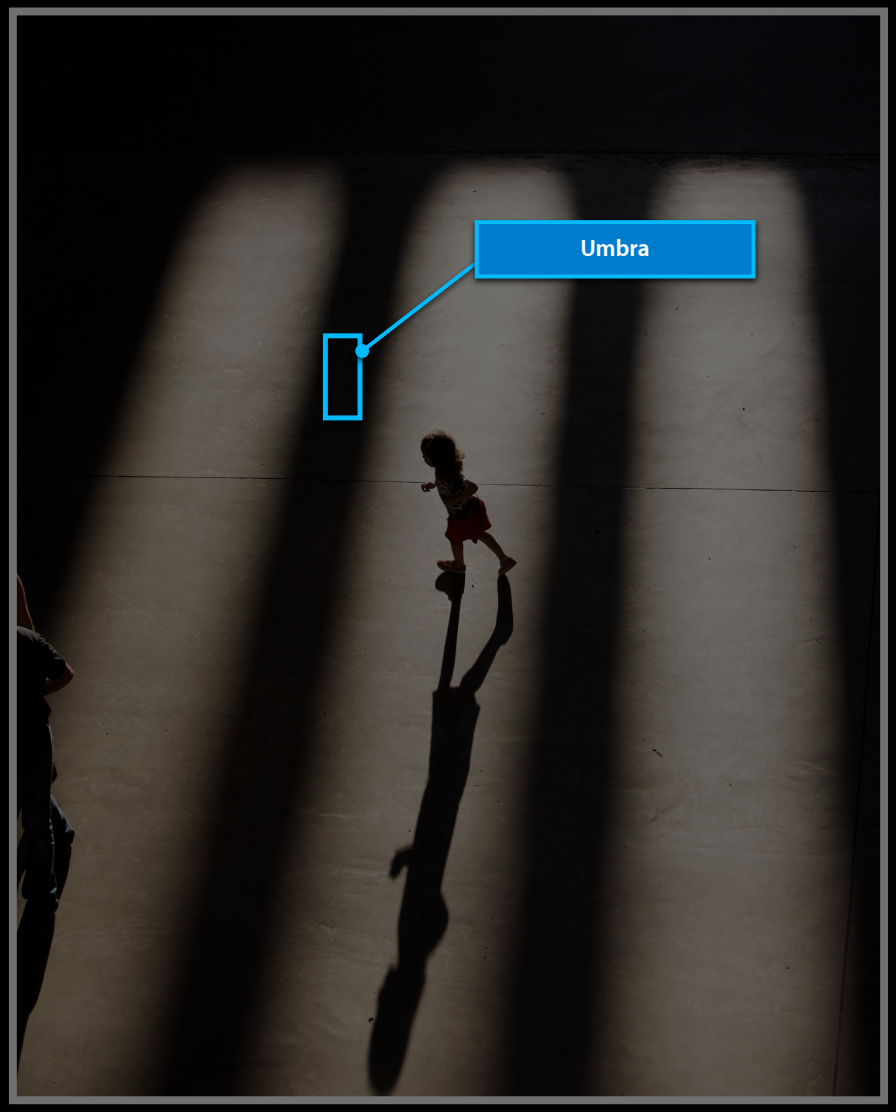
# Float Render Targets

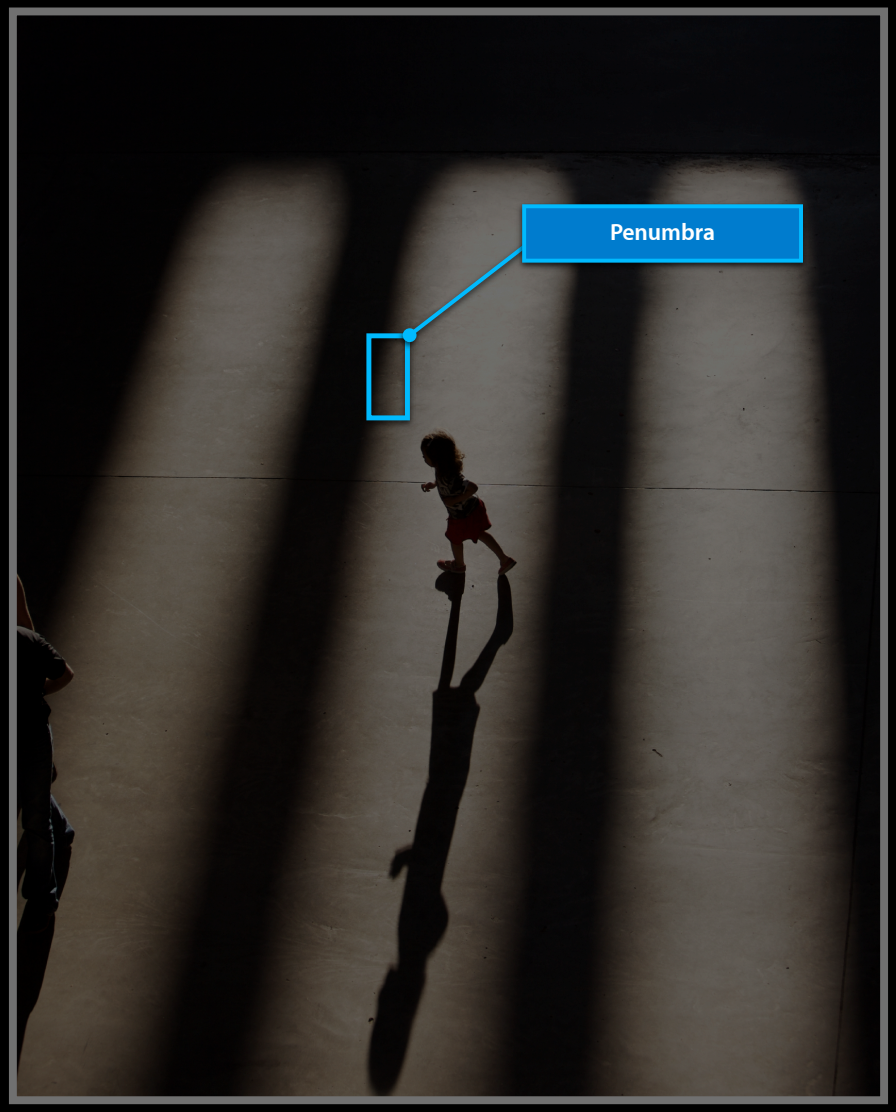
## APPLE_color_buffer_half_float

- Adds 16-bit floats as render targets
  - RGBA16F, RGB16F, RG16F, R16F
  - Clamping restrictions are relaxed
- iPad 2 only
- Multisampling is not supported

# Demo

# High-Quality Shadows


Umbra


Penumbra

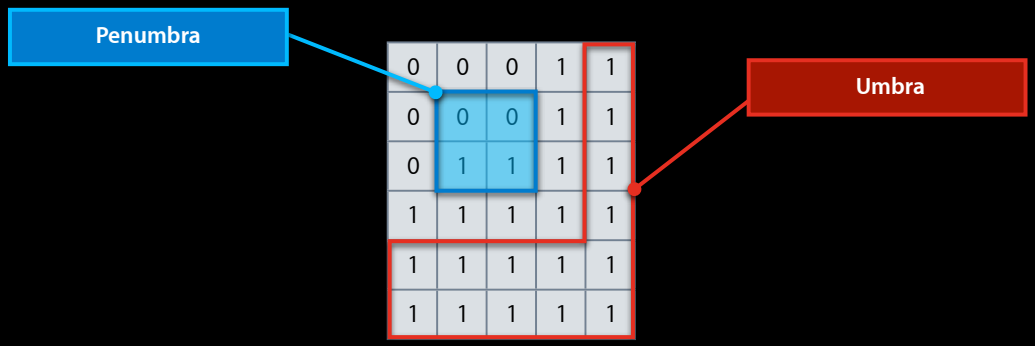
# Shadow Mapping with PCF

## APPLE_shadow_samplers

- Percentage Closer Filtering
  - Sample 4 values from depth texture
  - Compare
  - Return a weighted average
    - How much shadow contribution that pixel should receive
- iPad 2 only

# Shadow Mapping with PCF

## APPLE_shadow_samplers



# Shadow Mapping with PCF

## APPLE_shadow_samplers

- New GLSL defines

```
#extension GL_APPLE_shadow_samplers : require
```

- New GLSL sampler types

```
sampler2DShadow
```

- New GLSL functions

```
float shadow2DAPPLE(sampler2DShadow sampler, vec3 coord);
float shadow2DProjAPPLE(sampler2DShadow sampler, vec4 coord);
```

# Shadow Mapping with PCF

## APPLE_shadow_samplers

```
#extension GL_APPLE_shadow_samplers : require

uniform sampler2DShadow light_tex;
varying highp vec4 light_texcoord;
varying lowp vec3 color;
varying lowp float atten;

void main()
{
  highp float light_test = shadow2DProjAPPLE(light_tex, light_texcoord);
  lowp float clamp_atten = mix(atten, 1.0, light_test);
  gl_FragColor = vec4(color * clamp_atten, 1.0);
}
```

# Demo

# Enhancing the Performance

# Separate Shader Objects

# Separate Shader Objects

## Multiplicity of shaders

- Conventional
  - Vertex Shader + Fragment Shader -> Program
    - N Vertex Shaders + M Fragment Shaders
    - N*M Vertex Shader compilations
    - N*M Fragment Shader compilations
    - N*M Linkings

# Separate Shader Objects

## APPLE_separate_shader_objects

- APPLE_separate_shader_objects
  - Mix and Match Strategy with Program Pipeline Objects
  - N Vertex Programs + M Fragment Programs
    - N Vertex compilation + linking
    - M Fragment compilation + linking
    - N*M pairing (i.e., Program Pipeline Objects)

# Separate Shader Objects

## Pros and cons

- Shorter compilation and linking times
  - You can have more shaders for the same amount of overhead
  - Or faster start-up time
- Compiler loses the cross-stage optimization opportunities
  - Remove unused varyings
  - Dead-code elimination

# Separate Shader Objects

## Multiplicity of shaders

Compile/Link

Vertex Program

Compile/Link

Fragment Program 1

Compile/Link

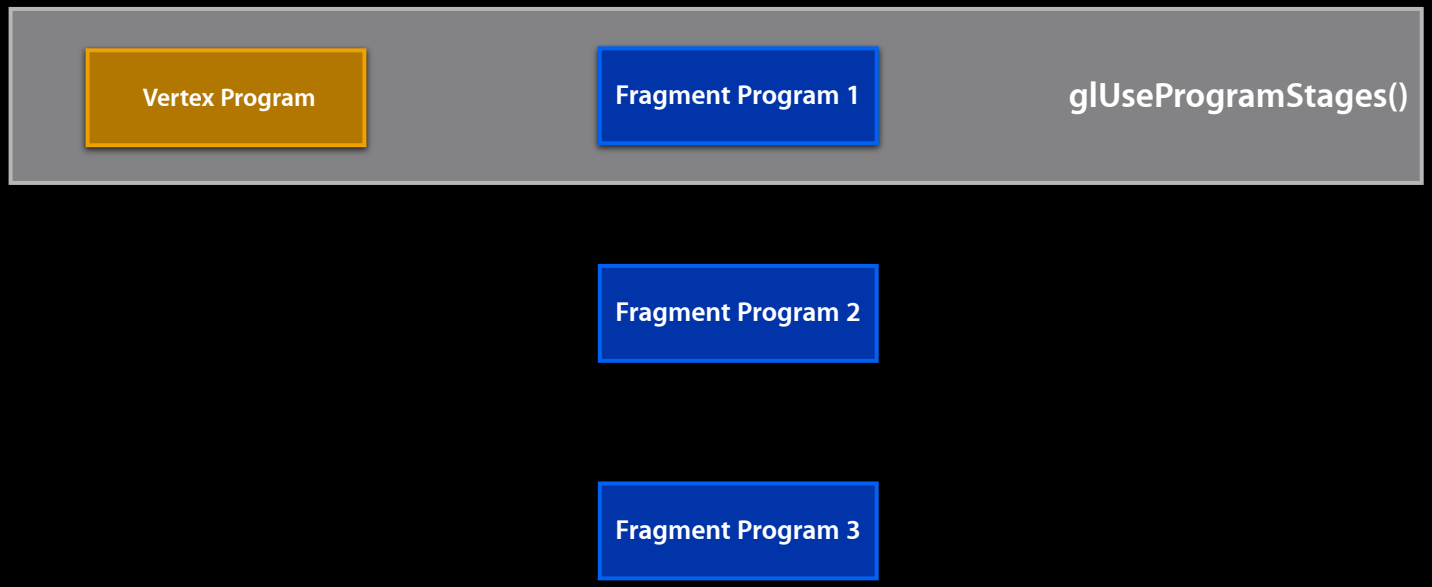
Fragment Program 2

Compile/Link

Fragment Program 3

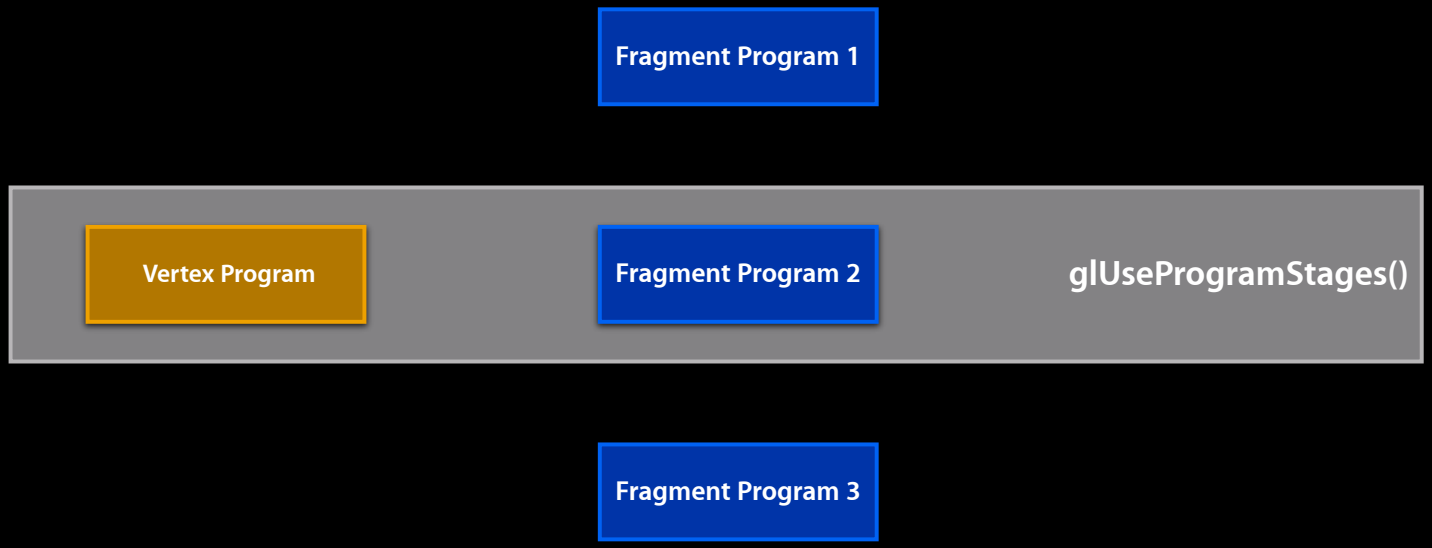
# Separate Shader Objects

## Multiplicity of shaders



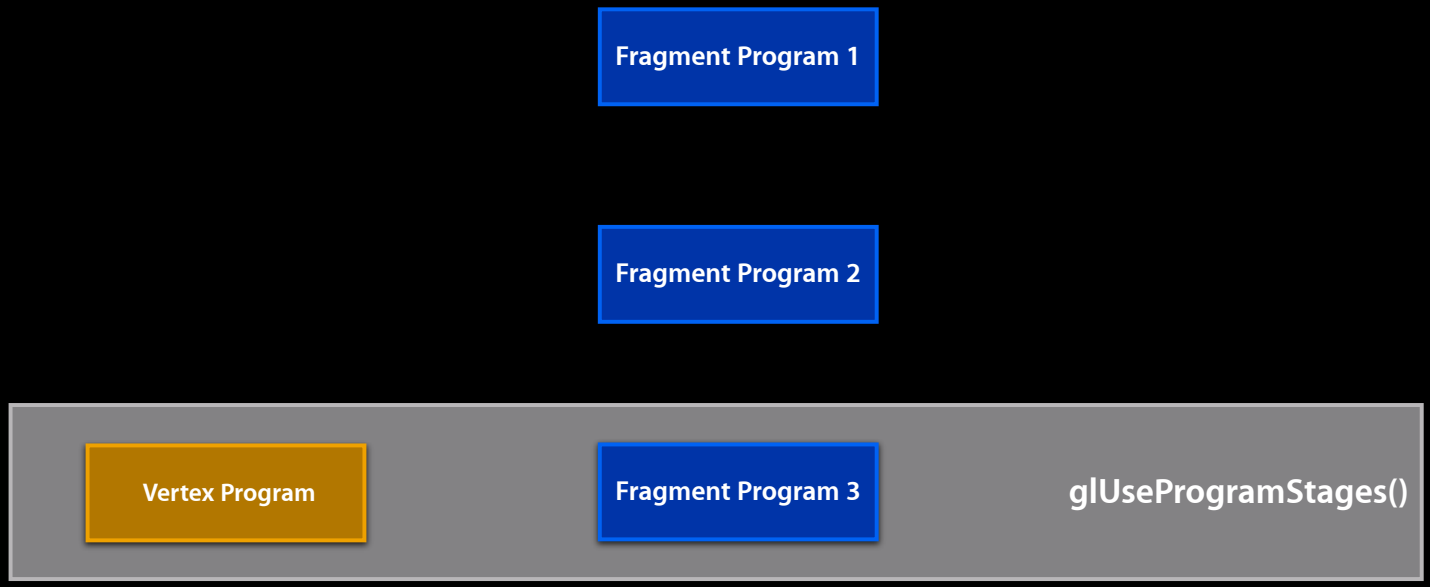
# Separate Shader Objects

## Multiplicity of shaders



# Separate Shader Objects

## Multiplicity of shaders



# Separate Shader Objects

## APPLE_separate_shader_objects

- Two usage models
  - Create program pipeline objects for each combination (faster)
  - Use the same program pipeline object and attach/detach program stages
- ARB_separate_shader_objects + ARB_explicit_attrib_location
  - Limited to ES-based constraints

# Occlusion Query

# Occlusion Query

## APPLE_occlusion_query_boolean

- Binary test to find if an object is visible
  - Current frame
    - Use bounding boxes in place of objects for query
  - Next frame
    - Check if any of the samples within the box are visible
    - If visible, draw the object
    - If not, do another query
  - Repeat
- iPad 2 only

# Occlusion Query

## Benefits

- Enhances performance by avoiding drawing
  - Better than depth based rejection
    - No vertex processing

# Occlusion Query

## APPLE_occlusion_query_boolean

```
//Frame N
   //render objects known to be visible

   // set a query and render a bounding box for the object
   glBeginQuery(gSamplesPassedTarget, queries[0]);
   {
    glBindVertexArrayOES(vao1);
    glUseProgram(..);
    glDrawArrays(GL_TRIANGLE_STRIP, ...);
   }

//Frame N+1
   //render objects known to be visible

   // check the query and use the results
   glGetQueryObjectiv(queries[0], GL_QUERY_RESULT_AVAILABLE, &ready);
   if(ready) {
     glGetQueryObjectuiv(queries[0], GL_QUERY_RESULT, &result);
     if(result) {
       //render the object
     }
   }
```

# New OpenGL ES Extensions

## Summary

- APPLE_texture_rg
- APPLE_color_buffer_half_float
- APPLE_shadow_samplers
- APPLE_separate_shader_objects
- APPLE_occlusion_query_boolean
- APPLE_debug_label
- APPLE_debug_marker
- OES_element_index_uint
- OES_texture_float_linear
- OES_texture_half_float_linear

# Related Sessions

| | |
|---|---|
| Tools for Tuning OpenGL ES Apps on iOS | Mission<br>Wednesday 3:15PM |
| Best Practices for OpenGL ES Apps in iOS | Mission<br>Wednesday 4:30PM |

# Labs

| | |
|---|---|
| OpenGL ES Lab | Graphics, Media & Games Lab A<br>Thursday 9:00AM |
| OpenGL ES Lab | Graphics, Media & Games Lab A<br>Thursday 2:00PM |

# 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


WWDC2011