Building a Game with SceneKit
Related Sessions

SceneKit basics

WWDC 2013
• "What's new in SceneKit"

WWDC 2014
• "What's new in SceneKit"
• "What's new in SpriteKit"
Outline

Where to start
How to give life to the scene
How we made the Bananas demo
How to get great performance
How you can build your own tools
Starting a Game
Displaying a 3D scene
Displaying a 3D Scene

Interface Builder
Displaying a 3D Scene

Interface Builder
Displaying a 3D Scene

Interface Builder
Displaying a 3D Scene

Interface Builder
Displaying a 3D Scene

Interface Builder
Xcode Template
Unified template for games

SceneKit
SpriteKit
GLEs
Managing Assets
SceneKit asset catalog

game.scnassets
Managing Assets

SceneKit asset catalog

Preserved file hierarchy
Organize models, textures, particles, etc
Automatic optimizations
Managing Assets
SceneKit asset catalog
Managing Assets

SceneKit asset catalog

Settings
- Automatically interleave geometry sources
- Always use the Y-up axis

Some documents use a different up axis. When enabled, objects in a scene are converted to the Y-up orientation.

iOS Image Settings
- Use PVRTC images when available

Use this option to prefer PVRTC to other file formats. When enabled, if an image is referenced by a scene and exists within the asset catalog with the same name and the .pvr file extension, the PVRTC version will be used by Scene Kit.
Managing Assets

SceneKit asset catalog

<table>
<thead>
<tr>
<th>Feature</th>
<th>iOS</th>
<th>OS X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up axis conversion</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Geometry interleaving</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>PVRTC textures</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>
Workflow

Working with artists

Artists

DAE

Developers
Workflow

Working with artists

Artists

DAE

Developers
Editing Assets

Scene editor
Editing Assets

Scene editor
Editing Assets

Scene editor

Direct manipulation
Editing Assets

Scene editor

Direct manipulation

Node hierarchy
Editing Assets

Scene editor

Direct manipulation
Node hierarchy
Editing Assets

Scene editor

Direct manipulation

Node hierarchy and entities
Editing Assets

Scene editor

Direct manipulation
Node hierarchy and entities
Node properties
Editing Assets

Scene editor

Direct manipulation
Node hierarchy and entities
Node properties
Editing Assets

Scene editor

Direct manipulation
Node hierarchy and entities
Node properties
Node attributes
Editing Assets

Scene editor

- Direct manipulation
- Node hierarchy and entities
- Node properties
- Node attributes
- Materials
Editing Assets

Particle system editor

Preview and edit

Particle system properties
Editing Assets

Particle system editor

Preview and edit

Particle system properties
Editing Assets

Particle system editor

Preview and edit

Particle system properties
Editing Assets

Particle system editor

Preview and edit

Particle system properties
Case Study

Bananas
Bananas

Sample code
Illustrates
• Animation
• Lighting and shadows
• Physics
• Particles
• Advanced rendering
Demo
Bananas
Bananas

Code base

All Objective-C code
Runs on iOS and OS X
~2700 lines of code
Level Design
Level Design
Level Design
# Level Design

## Statistics

<table>
<thead>
<tr>
<th>Features</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lights</td>
<td>10 lights in the world</td>
</tr>
<tr>
<td>Lights per object</td>
<td>3 lights (1 omni and 2 directional)</td>
</tr>
<tr>
<td>Polygons</td>
<td>200k polygons in the world</td>
</tr>
<tr>
<td>Polygons per frame</td>
<td>80k at most</td>
</tr>
<tr>
<td>Draw calls</td>
<td>50 at most</td>
</tr>
<tr>
<td>Textures</td>
<td>68M</td>
</tr>
</tbody>
</table>
Assets

Hierarchy

Asset catalogs
~25 3D documents
• Animations
• Models
• Textures
• Particle systems

game.scnassets
Managing Assets

Combining assets

Scene A + Scene B
Managing Assets

Combining assets

// Load two scenes
SCNScene *mainScene = [SCNScene sceneNamed:@"level.dae"];  
SCNScene *characterScene = [SCNScene sceneNamed:@"monkey.dae"];
Managing Assets
Combining assets

// Load two scenes
SCNScene *mainScene = [SCNScene sceneNamed:@"level.dae"];  
SCNScene *characterScene = [SCNScene sceneNamed:@"monkey.dae"];  

// Merge them
[mainScene.rootNode addChildNode:characterScene.rootNode];
Managing Assets

Combining assets

// Load two scenes
SCNScene *mainScene = [SCNScene sceneNamed:@"level.dae"];  
SCNScene *characterScene = [SCNScene sceneNamed:@"monkey.dae"];  

// Not allowed
[mainScene.rootNode addChildNode:characterScene.rootNode];
/ Load two scenes
SCNScene *mainScene = [SCNScene sceneNamed:@"level.dae"];
SCNScene *characterScene = [SCNScene sceneNamed:@"monkey.dae"];

// Get the monkey
SCNNode *monkey = [characterScene.rootNode childNodeWithName:@"monkey" recursively:YES];

// Add a monkey to the level
[mainScene.rootNode addChildNode:monkey];
Managing Assets

Combining assets

// Load two scenes
SCNScene *mainScene = [SCNScene sceneNamed:@"level.dae"];
SCNScene *characterScene = [SCNScene sceneNamed:@"monkey.dae"];

// Get the monkey
SCNNode *monkey = [characterScene.rootNode childNodeWithName:@"monkey"
    recursively:YES];

// Add a monkey to the level
[mainScene.rootNode addChildNode:monkey];

// Add another monkey to the level
[mainScene.rootNode addChildNode:[monkey clone]];
Controls
iOS

Support for game controllers
Custom “D-pad” gesture recognizer
• Three buttons
  - Jump
  - Left, right (with long press state)
Controls

OS X

Support for game controllers

Keyboard events

• Jump
• Left, right (with long press state)
Behaviors

Animating characters

skinned-character.dae
Behaviors

Animating characters

skinned-character.dae

run.dae  jump.dae  idle.dae
Behaviors

Animating characters

// Load an animation
CAAnimation *anim = [sceneSource entryWithIdentifier:animationName
withClass:[CAAnimation class]];
Behaviors

Animating characters

// Load an animation
CAAnimation *anim = [sceneSource entryWithIdentifier:animationName
               withObject:[CAAnimation class]];  

// Play it
[character addAnimation:anim
    forKey:@"run"];
Behaviors

Moving characters
Behaviors

Moving characters
Behaviors

Moving characters
Behaviors

Moving the camera

$c(0.0)$

$c(1.0)$
Behaviors

Moving the camera
Behaviors

Moving the camera
Collision Detection

Stay on the path
Avoid coconuts
Collect bananas
Jump over lava
Collision Detection

SCNPhysicsWorld

// Be notified through delegation
- (void)physicsWorld:(SCNPhysicsWorld *)world
didBeginContact:(SCNPhysicsContact *)contact;

// Or explicitly perform ray tests
- (NSArray *)rayTestWithSegmentFromPoint:(SCNVector3)origin
toPoint:(SCNVector3)dest
options:(NSDictionary *)options;
Behaviors

Animating items

// Animate a banana
[abanana runAction:]
    [SCNAction repeatActionForever:
        [SCNAction rotateByX:0.0
            y:2.0 * M_PI
            z:0.0
            duration:2.0]]];
Lighting
Lighting

Ambient
Lighting

Ambient
Lighting

Ambient
Shadows
Static shadows
Shadows
Static shadows
Shadows

Static shadows

Baked into textures
Don't adapt to changes

material.multiply.contents = shadowMap;
Shadows

Static shadows

Baked into textures
Don't adapt to changes

material.multiply.contents = shadowMap;
Shadows
Dynamic shadows
Shadows
Dynamic shadows
Shadows
Dynamic shadows

Shadow maps
Real-time shadows
Suitable for animated objects
Shadows
Dynamic shadows

Shadow maps
Real-time shadows
Suitable for animated objects

// Make a light cast shadows
light.castsShadow = YES;
Shadows

Mixing techniques

// Make the light cast shadows
aLight.castsShadow = YES;

// Exclude nodes that use baked shadows
aNode.castsShadow = NO;

// We can also use bit masks (with kMainLightMask & kSecondaryLightMask = 0)
aLight.categoryBitMask = kMainLightMask;
aNode.categoryBitMask = kSecondaryLightMask;
Shadows
Mixing techniques

// Make the light cast shadows
aLight.castsShadow = YES;

// Exclude nodes that use baked shadows
aNode.castsShadow = NO;

// We can also use bit masks (with kMainLightMask & kSecondaryLightMask = 0)
aLight.categoryBitMask = kMainLightMask;
aNode.categoryBitMask = kSecondaryLightMask;
Shadows
Mixing techniques

// Make the light cast shadows
aLight.castsShadow = YES;

// Exclude nodes that use baked shadows
aNode.castsShadow = NO;

// We can also use bit masks (with kMainLightMask & kSecondaryLightMask = 0)
aLight.categoryBitMask = kMainLightMask;
aNode.categoryBitMask  = kSecondaryLightMask;
Shadows
Mixing techniques

// Make the light cast shadows
aLight.castsShadow = YES;

// Exclude nodes that use baked shadows
aNNode.castsShadow = NO;

// We can also use bit masks (with kMainLightMask & kSecondaryLightMask = 0)
aLight.categoryBitMask = kMainLightMask;
aNNode.categoryBitMask = kSecondaryLightMask;
// Make the light cast shadows
aLight.castsShadow = YES;

// Exclude nodes that use baked shadows
aNode.castsShadow = NO;

// We can also use bit masks (with kMainLightMask & kSecondaryLightMask = 0)
aLight.categoryBitMask = kMainLightMask;
aNode.categoryBitMask = kSecondaryLightMask;
Shadows
Dynamic shadows

Shadow maps
Real-time shadows
Suitable for animated objects
Shadows
Projected shadows
Shadows
Projected shadows
Shadows
Projected shadows

Simplified shadows
Suitable for animated objects
Shadows
Projected shadows

// Use modulated mode
light.shadowMode = SCNShadowModeModulated;

// Configure the projected shadow
light.gobo.contents = aShadowImage;

// Use bit masks to specify receivers
light.categoryBitMask = kProjectorLightMask;
floor.categoryBitMask = kProjectorLightMask;
Particles

SCNParticleSystem

Used for torches and dust
Particles
SCNParticleSystem

// Load a particle system
SCNParticleSystem *particleSystem =
    [SCNParticleSystem particleSystemNamed:@"dust.scnp"
     inDirectory:@"art.scnassets/particles"];

// Attach to a node
[character addParticleSystem:particleSystem];
// Load a particle system
SCNParticleSystem *particleSystem =
    [SCNParticleSystem particleSystemNamed:@"dust.scnp"
     inDirectory:@"art.scnassets/particles"];

// Attach to a node
[character addParticleSystem:particleSystem];

// Control emission
particleSystem.birthRate = shouldEmit ? aBirthRate : 0;
Visual Improvements

Shader modifiers

Geometry animation
- Vines

Texture animation
- Lava
- Volcano
Visual Improvements

Shader modifiers

```swift
NSString *modifier = @"_geometry.texcoords[0] +=
    vec2(
        sin(_geometry.position.z + u_time) * 0.01,
        -0.05 * u_time
    );";

lavaNode.geometry.shaderModifiers =
    @{ SCNShaderModifierEntryPointGeometry : modifier};
```
Visual Improvements

Postprocessing

SCN Technique

• Color effects
• Image deformation
• Highly customizable
Visual Improvements

Postprocessing
Visual Improvements

SpriteKit overlays

iOS and OS X
SKScene
• Score
• Timer
• Title screen
Visual Improvements
SpriteKit overlays

Final Score

- Bananas: 75
- Coins: 6
- Total: 675
[SKAction **playSoundFileNamed:@"jump.caf"**

**waitForCompletion:NO**];
Performance Optimization

Thomas Goossens
Software Engineer
Performance Statistics
Xcode report
Performance Statistics
Xcode report
Performance Statistics
Xcode report

Each frame

-SceneKit applies constraints and renders the scene

-SceneKit simulates physics

-SceneKit evaluates animations

-didSimulatePhysics:

-update:

-didApplyAnimations:
Performance Statistics

Xcode report

Reports CPU usage
Target 16 ms per frame

Frame Time

- Animations: 3%
- Physics: 4%
- Constraints: 0%
- Particles: 4%
- Particles: 0%
- Rendering: 7%
- OpenGL Commands: 14%
- Idle: 68%

Graphics Report

228 MB
TEXTURES
Performance Statistics

Xcode report

Reports CPU usage
Target 16 ms per frame
Performance Statistics

In-view statistics

aSCNView.showsStatistics = YES;
Performance Statistics

In-view statistics

```
aSCNView.showsStatistics = YES;
```
Analyzing Performance Statistics

Bottleneck—CPU or GPU?
Analyzing Performance Statistics

Bottleneck—CPU or GPU?
Troubleshooting

Bottleneck—CPU

Less physic bodies
Less animations
Less actions
Less particles
Simpler game logic
Troubleshooting

Bottleneck—CPU

Less physic bodies
Less animations
Less actions
Less particles
Simpler game logic
Troubleshooting

Bottleneck—CPU

Less draw calls
Draw Calls

Number of draw calls is critical

CPU Impact
Draw Calls

Statistics
- Draw calls: 42
- Primitives: 300
- Vertices: 900
- Textures: 0
Draw Calls

**Draw calls**: 42
Draw Calls
Draw Calls
Flattening

Flatten to reduce the number of draw calls
Flatten directly in 3D tools (recommended)
Flatten programmatically

// Flatten a node tree
SCNNode *flattenedNode = [node flattenedClone];
Flattening

Don’t flatten too much
Verify total number of polygons per frame
Verify number of influencing lights
Flattening

Flattening and culling in Bananas
Troubleshooting

Bottleneck—GPU

Tiler
Renderer/device
Troubleshooting

Bottleneck—GPU

Tiler

Renderer/device
Troubleshooting
Bottleneck—Renderer/device

Fill rate limited
Fragment shaders limited

Problems & Solutions
- Performance limited by Fragment Shading
  - The program with the most costly fragment shader is:
    - Program #3 "Material"
  - For best performance, render all opaque geometry before any transparent geometry.
    - 89 glDrawElements (GL_TRIANGLES, 60, GL_UNSIGNED_SHORT, nullptr)
    - 102 glDrawElements (GL_TRIANGLES, 60, GL_UNSIGNED_SHORT, nullptr)
  - The fragment shaders in these programs perform dependent texture reads, which are slower than non-dependent texture reads.
Fill Rate
Fill Rate

Contents scale factor
• 1x, 2x, intermediate
Fill Rate

Contents scale factor
• 1x, 2x, intermediate

Reduce screen space postprocesses
• Deferred shadows, depth of field, reflective floor
• Custom techniques
Fill Rate

Contents scale factor
• 1x, 2x, intermediate

Reduce screen space postprocesses
• Deferred shadows, depth of field, reflective floor
• Custom techniques

Anti-aliasing

//msaa 4x
aSCNView.antialiasingMode = SCNAntialiasingModeMultisampling4X;
//no antialiasing
aSCNView.antialiasingMode = SCNAntialiasingModeNone;
# Lighting

## Dynamic vs. static

<table>
<thead>
<tr>
<th></th>
<th>Dynamic Lighting</th>
<th>Static Lighting</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Quality</strong></td>
<td>Good</td>
<td>Best</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>Costly</td>
<td>Fast</td>
</tr>
<tr>
<td>Work with animations</td>
<td>✔️</td>
<td></td>
</tr>
</tbody>
</table>
Lighting

Area of influence

aLight.attenuationEndDistance = 100.0;
Lighting

Area of influence

aLight.attenuationEndDistance = 100.0;
Lighting
Influencing lights

Number of influencing lights per object is what matters
Lighting

Influencing lights

Number of influencing lights per object is what matters
Lighting

Banana

Omni lights located far apart
Objects lit by 1 omni light at maximum
Lighting

Banana

Omni lights located far apart
Objects lit by 1 omni light at maximum
Lighting
Optimizing shadows

Dynamic

Static
Lighting
Optimizing shadows

Dynamic

Static
# Shadows

Dynamic shadow modes

<table>
<thead>
<tr>
<th></th>
<th>Forward</th>
<th>Projected</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>API</strong></td>
<td>SCNShadowMode<strong>Forward</strong></td>
<td>SCNShadowMode<strong>Modulated</strong> + gobo image</td>
</tr>
<tr>
<td><strong>Performance impact</strong></td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Quality</strong></td>
<td>Best</td>
<td>Low</td>
</tr>
</tbody>
</table>
Shadows

Optimizing shadows

// Set the size of the shadow map
aLight.shadowMapSize = CGSizeMake(512, 512);
Shadows

Optimizing shadows

// Set the size of the shadow map
aLight.shadowMapSize = CGSizeMake(512, 512);
Shadows

Optimizing shadows

// Configure the number of sample in the shadow map
aLight.shadowSampleCount = 1.0;
Shadows
Optimizing shadows

```java
// Configure the number of sample in the shadow map
aLight.shadowSampleCount = 1.0;
```
Texturing

Avoid unnecessarily large textures
Pack textures into texture atlases
Texturing

Mipmapping

Advantages
• Can improve performance significantly
• Reduce aliasing/Moiré effect

Drawbacks
• Higher load time
• ~30% more memory

// Turn on mipmapping
aMaterial.diffuse.mipFilter = SCNFilterModeLinear;
Polygon Rate

Problems & Solutions

- **Performance limited by Fragment Shading**
  - The program with the most costly fragment shader is:
    - Program #3 "Material"
  - For best performance, render all opaque geometry before any transparent geometry.
    - 89 glDrawElements(GL_TRIANGLES, 60, GL_UNSIGNED_SHORT, nullptr)
    - 102 glDrawElements(GL_TRIANGLES, 60, GL_UNSIGNED_SHORT, nullptr)
  - The fragment shaders in these programs performed dependent texture reads, which are slower than non-dependent texture reads.
    - Program #3 "Material"
    - Program #5 "Fairy"
  - Here are some additional things you can do that may improve performance:
    - Reduce the length of your fragment shaders.
    - Use a simpler algorithm. For example, use textures as lookup tables to replace complex functions.
    - Avoid using higher precision levels than necessary. For example, you only need to use float for RGBA texture data.
    - Move work that can be linearly interpolated to vertex shader.

- **Performance limited by Texture Filtering in Fragment Shaders**
  - Performance is limited by performing filtering on texture reads in a fragment shader. Use mipmapped bilinear.
Polygon Rate

Problems & Solutions

Performance limited by Fragment Shading

- The program with the most costly fragment shader is:
  - Program #3 "Material"

- For best performance, render all opaque geometry before any transparent geometry.
  - 89 glDrawElements(GL_TRIANGLES, 60, GL_UNSIGNED_SHORT, nullptr)
  - 102 glDrawElements(GL_TRIANGLES, 60, GL_UNSIGNED_SHORT, nullptr)

- The fragment shaders in these programs performed dependent texture reads, which are slower than non-dependent texture reads.
  - Program #3 "Material"
  - Program #5 "Fairy"

Here are some additional things you can do that may improve performance:
- Reduce the length of your fragment shaders.
- Use a simpler algorithm. For example, use textures as lookup tables to replace complex functions.
- Avoid using higher precision levels than necessary. For example, you only need to use lowp for RGB565 texture data.
- Move work that can be linearly interpolated to vertex shader.

Performance limited by Texture Filtering in Fragment Shaders

Performance is limited by performing filtering on texture reads in a fragment shader. Use mipmapped bilinear filtering.
Polygon Rate

Graphics Report

Frame Time
- Animations: 3%
- Physics: 4%
- Constraints: 0%
- Particles: 4%
- Delegate: 0%
- Rendering: 7%
- OpenGL Commands: 14%
- Idle: 58%

Memory
- 228 MB (Textures)
- 55 MB (Geometry)
- 107 MB (FBO Buffers)

Statistics
- Draw calls: 1
- Primitives: 13700
- Vertices: 900
- Textures: 0
Polygon Rate

Graphics Report

Frame Time

- Animations: 3%
- Physics: 4%
- Constraints: 0%
- Particles: 4%
- Delegate: 0%
- Rendering: 7%
- OpenGL Commands: 14%
- Idle: 98%

Memory

- 228 MB: TEXTURES
- 55 MB: GEOMETRY
- 107 MB: Framebuffers

Statistics

- Primitives: 13700
- Vertices: 330000
- Textures: 0

No issues
Polygon Rate
Polygon Rate
Levels of Detail

Reduce polygon count per frame

```swift
SCNLevelOfDetail *lod1 = [SCNLevelOfDetail
    levelOfDetailWithGeometry:aGeometry
    worldSpaceDistance:aDistance];
geometry.levelOfDetail = @[lod1, lod2, ...];
```
Levels of Detail

Reduce polygon count per frame

```swift
SCNLevelOfDetail *lod1 = [SCNLevelOfDetail levelOfDetailWithGeometry:aGeometry worldSpaceDistance:aDistance];
geometry.levelOfDetail = @[lod1, lod2, ...];
```
Troubleshooting

Summary

Identify bottlenecks

- CPU
- Tiler
- Renderer or device
Troubleshooting

Bottleneck—CPU

Reduce draw calls by flattening
Less physic bodies
Less animations
Less actions
Troubleshooting

Bottleneck—Tiler

Levels of detail
Split scenes in chunks
Troubleshooting

Bottleneck—Renderer or device

Simpler materials
Less/simpler lights
Smaller textures, mipmapping
Downscaled contents size
Less postprocess
No multisampling
Performance Notes
// Duplicate a node
SCNNode *nodeB = [nodeA copy];
// Duplicate a node
SCNNode *nodeB = [nodeA copy];
// Duplicate a node
SCNNode *nodeB = [nodeA copy];
// Duplicate a node
SCNNode *nodeB = [nodeA copy];
// Duplicate a node
SCNNNode *nodeB = [nodeA copy];

// Un-share the geometry (cheap)
nodeB.geometry = [nodeA.geometry copy];
// Duplicate a node
SCNNode *nodeB = [nodeA copy];

// Un-share the geometry (cheap)
nodeB.geometry = [nodeA.geometry copy];

// Un-share the material
nodeB.geometry.firstMaterial = [nodeA.geometry.firstMaterial copy];
Sharing

// Duplicate a node
SCNNNode *nodeB = [nodeA copy];

// Un-share the geometry (cheap)
nodeB.geometry = [nodeA.geometry copy];

// Un-share the material
nodeB.geometry.firstMaterial = [nodeA.geometry.firstMaterial copy];
Preloading

Supported by SCNView, SCNLayer and SCNRenderer

- (void)prepareObjects:(NSArray *)objects
  withCompletionHandler:(BOOL (^)(BOOL success))completionHandler;
## Preloading

<table>
<thead>
<tr>
<th>Object</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCNMaterial</td>
<td>Textures</td>
</tr>
<tr>
<td>SCNGeometry</td>
<td>Geometry buffers and materials</td>
</tr>
<tr>
<td>SCNNode</td>
<td>Node tree and geometries</td>
</tr>
<tr>
<td>SCNScene</td>
<td>Node tree and shaders (lighting)</td>
</tr>
</tbody>
</table>
Creating Custom Tools
Archiving

Adopted by SCNScene, SCNNode, SCNGeometry, SCNMaterial, etc.
Conform to NSSecureCoding

// Archive a scene
NSData *archivedScene = [NSKeyedArchiver archivedDataWithRootObject:scene];

// Unarchive a scene
SCNScene *scene = [NSKeyedUnarchiver unarchiveObjectWithData:archivedScene];
Exporting

Export to Collada
Import back in 3D software
Only a subset of SceneKit can be exported

- 
  (BOOL)writeToURL:(NSURL *)url
  options:(NSDictionary *)options
  delegate:(id <SCNSceneExportDelegate>)delegate
  progressHandler:(SCNSceneExportProgressHandler)progressHandler;
# Archiving vs. Exporting

<table>
<thead>
<tr>
<th>Archiving</th>
<th>Collada export</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loadable on iOS</td>
<td></td>
</tr>
<tr>
<td>Can be read by DCC tools</td>
<td></td>
</tr>
<tr>
<td>Scene graph (nodes, geometry, materials…)</td>
<td></td>
</tr>
<tr>
<td>Animations</td>
<td></td>
</tr>
<tr>
<td>Actions</td>
<td></td>
</tr>
<tr>
<td>Physics</td>
<td></td>
</tr>
<tr>
<td>Particles</td>
<td></td>
</tr>
<tr>
<td>Custom shaders</td>
<td></td>
</tr>
</tbody>
</table>
### Archiving vs. Exporting

<table>
<thead>
<tr>
<th>Archiving</th>
<th>Collada export</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loadable on iOS</td>
<td>Through Xcode</td>
</tr>
<tr>
<td>Can be read by DCC tools</td>
<td></td>
</tr>
<tr>
<td>Scene graph (nodes, geometry, materials…)</td>
<td></td>
</tr>
<tr>
<td>Animations</td>
<td></td>
</tr>
<tr>
<td>Actions</td>
<td></td>
</tr>
<tr>
<td>Physics</td>
<td></td>
</tr>
<tr>
<td>Particles</td>
<td></td>
</tr>
<tr>
<td>Custom shaders</td>
<td></td>
</tr>
</tbody>
</table>
# Archiving vs. Exporting

<table>
<thead>
<tr>
<th></th>
<th>Archiving</th>
<th>Collada export</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loadable on iOS</td>
<td>✓</td>
<td>Through Xcode</td>
</tr>
<tr>
<td>Can be read by DCC tools</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Scene graph</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(nodes, geometry, materials…)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Animations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Particles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Custom shaders</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Archiving vs. Exporting

<table>
<thead>
<tr>
<th></th>
<th>Archiving</th>
<th>Collada export</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loadable on iOS</td>
<td>✓</td>
<td>Through Xcode</td>
</tr>
<tr>
<td>Can be read by DCC tools</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Scene graph</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(nodes, geometry, materials…)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Animations</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Actions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Particles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Custom shaders</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Archiving vs. Exporting

<table>
<thead>
<tr>
<th></th>
<th>Archiving</th>
<th>Collada export</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loadable on iOS</td>
<td>✔</td>
<td>Through Xcode</td>
</tr>
<tr>
<td>Can be read by DCC tools</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Scene graph (nodes, geometry, materials…)</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Animations</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Actions</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Physics</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Particles</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Custom shaders</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
Scripting

JavaScript bridge
SceneKit is fully scriptable

void SCNExportJavaScriptModule(JSContext *context);
Code Samples

Sample code “Bananas”
Checkout the three other code samples
More Information

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

Filip Iliescu
Graphics and Game Technologies Evangelist
filiescu@apple.com

Documentation
SceneKit Framework Reference
http://developer.apple.com

Apple Developer Forums
http://devforums.apple.com
## Related Sessions

<table>
<thead>
<tr>
<th>Session Type</th>
<th>Location</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s New in SpriteKit</td>
<td>Pacific Heights</td>
<td>Wednesday 2:00PM</td>
</tr>
<tr>
<td>What’s New in SceneKit</td>
<td>Pacific Heights</td>
<td>Thursday 10:15AM</td>
</tr>
<tr>
<td>Labs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• SceneKit Lab</td>
<td>Graphics and Games Lab A</td>
<td>Thursday 12:45PM</td>
</tr>
</tbody>
</table>