Advances in SceneKit Rendering

Session 609

Amaury Balliet SceneKit Engineer
Jean-Baptiste Bégué SceneKit Engineer
Sébastien Métrot SceneKit Engineer
Nick Porcino Model I/O Engineer
Agenda

SceneKit in a Nutshell
Rendering Advances
Demo
Behind the Demo
Camera Effects
Model I/O
SceneKit
In a nutshell

Amaury Balliet SceneKit Engineer
SceneKit

Challenge: Use the AND, OR, and NOT operators to navigate Byte through the world.

Each of these operators influences the way your conditional code runs:

- The NOT operator (!) inverts a Boolean value, saying, "if NOT this condition, do this."
- The AND operator (&&) combines two conditions and runs the code only if both are true.
- The OR operator (||) combines two conditions and runs the code if at least one is true.

Solve the challenge by choosing the operators that will work best so that Byte collects all the gems and toggles open the switches.

```swift
for i in 1...6 {
    moveForward()
    if isOnClosedSwitch && isBlocked {
        toggleSwitch()  
        turnLeft()     
    }
    moveForward()
}
```
ATP structure
ATP is a multipurpose storehouse of chemical energy that can be used by cells in a variety of reactions. Although other carrier molecules transport energy within cells, ATP is the most abundant energy-carrier molecule in cells and is found in all types of organisms. As shown in Figure 4, ATP is a nucleotide made of an adenine base, a ribose sugar, and three phosphate groups.
SceneKit
Thank you!
macOS   iOS
macOS  iOS  tvOS
macOS  iOS  tvOS  watchOS
SceneKit
watchOS 3

SceneKit is now available everywhere
Great opportunity to make attractive apps
New interactions with content on your wrist
SceneKit
watchOS 3

SceneKit is now available everywhere
Great opportunity to make attractive apps
New interactions with content on your wrist
SceneKit
watchOS 3

SceneKit is now available everywhere
Great opportunity to make attractive apps
New interactions with content on your wrist

Game Technologies for Apple Watch

Mission
Friday 3:00PM
<table>
<thead>
<tr>
<th>SceneKit</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s New in SceneKit</td>
<td>WWDC 2013</td>
</tr>
<tr>
<td>What’s New in SceneKit</td>
<td>WWDC 2014</td>
</tr>
<tr>
<td>Enhancements to SceneKit</td>
<td>WWDC 2015</td>
</tr>
</tbody>
</table>
Advances in SceneKit Rendering
Physically based rendering
Physically based rendering in the hands of everyone.
Advances in SceneKit Rendering

Biggest leap forward since SceneKit’s introduction

Latest advances in 3D graphics

Modern technologies

• Accurate rendering

• Physically based materials and lighting
Accurate Rendering
Linear Rendering and Color Management

Linear rendering
Linear Rendering and Color Management

Linear rendering
Linear Rendering and Color Management

Linear rendering
Linear Rendering and Color Management

Linear rendering
Linear Rendering and Color Management

Linear rendering
Linear Rendering and Color Management

Shading in gamma space
Linear Rendering and Color Management

Shading in linear space

Textured → Gamma Decoding → Shading → Gamma Encoding → Texture or Framebuffer

Shading in linear space
Linear Rendering and Color Management

Shading in gamma space
Linear Rendering and Color Management

Shading in linear space
Linear Rendering and Color Management

Linear rendering
Linear Rendering and Color Management

Linear rendering

Essential for physically based shading

Being linear is necessary to get the math right

Benefits to all other lighting models
Linear Rendering and Color Management

Color management

Cross-framework effort for color accuracy

Fully embraced by SceneKit
Linear Rendering and Color Management

Color management for textures

Automatic color management for images
Textures that represent raw data are supposed to be sRGB
Have a look at texture sets and asset catalogs
Linear Rendering and Color Management

Color management for textures

Automatic color management for images
Textures that represent raw data are supposed to be sRGB
Have a look at texture sets and asset catalogs
Color management for textures

Automatic color management for images
Textures that represent raw data are supposed to be sRGB
Have a look at texture sets and asset catalogs
Linear Rendering and Color Management

Color management for color objects

Automatic color management for color objects
Color components previously assumed to be sRGB
Be careful with programmatically-generated color objects
let colorA = NSColor(displayP3Red: 0.5, green: 1.0, blue: 0.75, alpha: 1)  // Display P3
let colorB = NSColor(srgbRed: 0.5, green: 1.0, blue: 0.75, alpha: 1)      // sRGB
Linear Rendering and Color Management

Color management for color objects
Linear Rendering and Color Management

Color management for color objects

Automatic color management for color objects
Color components previously assumed to be sRGB
Be careful with programmatically-generated color objects
Be careful with shader modifiers

// Metal Shading Language shader modifier
// linear extended sRGB components for sRGB(0.5, 1.0, 0.75)
_surface.diffuse.rgb += float3(0.235514164, 1.03112769, 0.523271978)
Linear Rendering and Color Management

Backward compatibility

No performance cost

Enabled when building against the new SDKs

Dramatic visual impact for older scenes
Linear Rendering and Color Management

Backward compatibility
Linear Rendering and Color Management

Backward compatibility
Linear Rendering and Color Management

Backward compatibility
Linear Rendering and Color Management

Backward compatibility

No performance cost
Enabled when building against the new SDKs
Dramatic visual impact for older scenes
Global option to opt-out

// Info.plist
<key>SCNDisableLinearSpaceRendering</key>
<true/>
Wide Gamut Content
Wide Gamut Content

Transparent support for wide gamut images and color

Full support of wide gamut displays

- 9.7-inch iPad Pro
- iMac with Retina display
Wide Gamut Content

Caveats

Increased memory usage
Global option to opt-out

// Info.plist
<key>SCNDisableWideGamut</key>
<true/>
Wide Gamut Content

“Color Gamut Showcase” sample code
Wide Gamut Content

Working with Wide Color

Mission

Thursday 1:40PM
Advances in SceneKit Rendering

Biggest leap forward since SceneKit’s introduction
Latest advances in 3D graphics
Modern technologies
• Accurate rendering
• Physically based materials and lighting
Physically Based Rendering
Physically Based Rendering
Physically Based Rendering
Physically Based Rendering
Physically Based Rendering

Bidirectional reflectance distribution function

\[ L_o(v) = \int_\Omega f(l, v) L_t(l) \langle n \cdot l \rangle dl \]

\[ f(l, v) = f_d(l, v) + f_r(l, v) \]

\[ f_d(l, v) = \frac{c_{\text{diff}}}{\pi} \quad f_r(l, v) = \frac{D(h) G(l, v) F(l, v)}{4 \langle n \cdot l \rangle \langle n \cdot v \rangle} \]
Physically Based Rendering

Relies on intuitive physical material properties
Adopted and loved by artists
High-level and easy-to-use API
Physically Based Rendering

Physically based materials
Physically based lights
Physically Based Materials
Physically Based Materials
Physically Based Materials
Physically Based Materials

Diffuse Reflection
Specular Reflection
Physically Based Materials

Diffuse reflection
Physically Based Materials

Diffuse reflection
Physically Based Materials

Specular reflection
Physically Based Materials

Specular reflection
Physically Based Materials

Specular reflection
Physically Based Materials

Reflectance
Physically Based Materials

Reflectance

Plastic  Aluminum  Gold
public class SCNMaterial {
    public var metalness: SCNMaterialProperty { get }
}

Physically Based Materials
Metalness
Physically Based Materials

Metal versus dielectric

<table>
<thead>
<tr>
<th>Metal</th>
<th>Dielectric</th>
</tr>
</thead>
<tbody>
<tr>
<td>High reflectance</td>
<td>Low reflectance</td>
</tr>
<tr>
<td>Absorb light</td>
<td>Absorb and scatter light</td>
</tr>
</tbody>
</table>
# Physically Based Materials

## Metal versus dielectric

<table>
<thead>
<tr>
<th>Metal</th>
<th>Dielectric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bright specular reflection</td>
<td>Specular reflection at grazing angles</td>
</tr>
<tr>
<td>No diffuse reflection</td>
<td>Mainly diffuse reflection</td>
</tr>
</tbody>
</table>
Physically Based Materials

Metal versus dielectric

<table>
<thead>
<tr>
<th>Metal</th>
<th>Dielectric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reflectance at 0° diffuse map</td>
<td>Reflectance at 0° constant</td>
</tr>
<tr>
<td>Object albedo diffuse map</td>
<td>Object albedo diffuse map</td>
</tr>
</tbody>
</table>
Physically Based Materials

Metal versus dielectric

```java
public class SCNMaterial {
    public var diffuse: SCNMaterialProperty { get }
}
```
Physically Based Materials

Roughness
Physically Based Materials

Roughness
public class SCNMaterial {
    public var roughness: SCNMaterialProperty { get }
}

Physically Based Materials

Roughness
Physically Based Materials

Material API

Three fundamental properties
- Albedo or reflectance at 0°
- Metalness
- Roughness

```csharp
public class SCNMaterial {
    public var diffuse: SCNMaterialProperty { get }
    public var metalness: SCNMaterialProperty { get }
    public var roughness: SCNMaterialProperty { get }
}
```
Physically Based Materials

Material API

New physically based lighting model

diffuse, metalness, and roughness maps

```swift
let material = SCNMaterial()
material.lightingModelName = .physicallyBased
material.diffuse.contents = "albedo.png"
material.metalness.contents = "metalness.png"
material.roughness.contents = "roughness.png"
```
diffuse map, roughness map
diffuse map, metalness map
diffuse map, metalness map, roughness map
Physically Based Materials

Material API

Use grayscale images for metalness, roughness, and ambientOcclusion.

Use scalars for constant values:

```swift
material.metalness.contents = "metalness.png"
material.roughness.contents = NSNumber(value: 0.5)
```
Physically Based Materials
Physically Based Rendering

Physically based materials
Physically based lights
Physically Based Lights
Physically Based Lights

Image based lighting
Light probes
Point lights
Physically Based Lights

Image based lighting
Light probes
Point lights
Physically Based Lights

Image based lighting
Physically Based Lights

Image based lighting
Physically Based Lights

Image based lighting
Physically Based Lights
Image based lighting

Cube map captures the environment
Lighting information is derived from cube map
Image based lighting can be used alone
Not mandatory to add lights in the scene
Physically Based Lights

Image based lighting

A single change affects the whole scene

```swift
let scene = SCNScene()
scene.lightingEnvironment.contents = "outside.exr"
```
Physically Based Lights

Image based lighting

A single change affects the whole scene

Works great with the `background` property

```swift
let scene = SCNScene()
scene.lightingEnvironment.contents = "outside.exr"
scene.background.contents = scene.lightingEnvironment.contents
```
Physically Based Lights

Image based lighting: Caveats

- Captures the distant environment
- Does not account for obstacles in the scene
- Not suited for occluded objects
Physically Based Lights

Image based lighting
Light probes
Point lights
Physically Based Lights

Light probes
Physically Based Lights

Light probes
Physically Based Lights

Light probes

A special kind of light
Captures the local diffuse lighting
Account for obstacles in the scene
Lightweight
Efficient
Physically Based Lights

Light probes

A special kind of light: **SCNLightType.probe**

```swift
let light = SCNLight()
light.type = .probe
```
Physically Based Lights

Light probes

Can be placed programmatically or in Xcode
Static lighting information must be baked

```swift
public class SCNRenderer {
    public func updateProbes(_ probes: [SCNNode], atTime time: CFTimeInterval)
}
```
Physically Based Lights

Image based lighting
Light probes
Point lights
Physically Based Lights

Point lights

Work with physically based materials, too

Updated to be configured with real-world properties

```swift
public let SCNLightTypeOmni: String        // Omnidirectional light
public let SCNLightTypeDirectional: String // Directional light
public let SCNLightTypeSpot: String        // Spot light
```
Physically Based Lights

Point lights: Intensity

Expressed in lumens (lm)

```swift
let light = SCNLight()
light.intensity = 1500 // defaults to 1000 lm
```
Physically Based Lights

Point lights: Temperature

Expressed in Kelvin (K)
Modulates the light’s color

```swift
let light = SCNLight()
light.temperature = 5000
```
Physically Based Lights

Photometric lights

New kind of point light

<table>
<thead>
<tr>
<th>Public let</th>
<th>SCNLightTypeOmni: String</th>
<th>// Omnidirectional light</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public let</td>
<td>SCNLightTypeDirectional: String</td>
<td>// Directional light</td>
</tr>
<tr>
<td>Public let</td>
<td>SCNLightTypeSpot: String</td>
<td>// Spot light</td>
</tr>
<tr>
<td>Public let</td>
<td>SCNLightTypeIES: String</td>
<td>// IES light</td>
</tr>
</tbody>
</table>
Physically Based Lights

Photometric lights

New kind of point light

```swift
public let SCNLightTypeOmni: String        // Omnidirectional light
public let SCNLightTypeDirectional: String // Directional light
public let SCNLightTypeSpot: String        // Spot light
public let SCNLightTypeIES: String         // IES light
```
Physically Based Lights

Photometric lights

New kind of point light
Modeled after real-world lights
Custom attenuation shape
Physically Based Lights

Photometric lights

Spot

IES
Physically Based Lights

Photometric lights

Spot

IES
Physically Based Lights

Photometric lights

New kind of point light
Modeled after real-world lights
Custom attenuation shape

```swift
let light = SCNLight()
light.type = .IES
light.iesProfileURL = Bundle.main().urlForResource("spot", withExtension: "ies")
```
Physically Based Rendering

Recap

Physically based materials

Advanced lighting

- Image based lighting
- Light probes
- Point lights
Demo

Jean-Baptiste Bégué SceneKit Engineer
Sébastien Métrot SceneKit Engineer
Physically Based Rendering in Practice

Bob the Badger

Sébastien Métrot SceneKit Engineer
The demo
The demo is a sample code!
Pre-Production

Drafts
Pre-Production

Drafts
Production
Modeling
Workflow

Our artist exported models and PBR materials as DAE files

Custom tool written in SceneKit

- Import DAE file
- Convert units to meters
- Add light probes along the track
Lighting

Image based lighting

Light coming from the environment

Great for outdoor scenes

Reflections

Works with regular lights, too
Lighting

Image based lighting

Light coming from the environment
Great for outdoor scenes
Reflections
Works with regular lights, too

Background Image
Lighting

Image based lighting

Light coming from the environment
Great for outdoor scenes
Reflections
Works with regular lights, too

Background Image

Lighting Environment
Lighting

Light probes

Custom tool adds light probes along a path.
They can be placed and computed in Xcode.
Essential for the inside.
May be optional for an outside-only scene.
Lighting

Light probes

Custom tool adds light probes along a path
They can be placed and computed in Xcode
Essential for the inside
May be optional for an outside-only scene
Lighting
Light maps

For the inside
Overrides IBL except for the specular component

```swift
let material = SCNMaterial()
matterial.selfIllumination.contents = "selfIllum.exr"
```
Normal maps add detail to the models

```swift
let material = SCNMaterial()
material.normal.contents = "normal.png"
```
Ambient occlusion maps make global illumination more realistic.

```swift
let material = SCNMaterial()
material.ambientOcclusion.contents = "ao.png"
```
Point lights

One global dynamic light high above the scene

- Create shadows
- Improve global lighting
Materials

100% physically based materials!
Materials

Properties
- Lighting model: Physically Based
- Diffuse
- Metalness: Metallic
- Roughness: Float value
- Normal
- Occlusion
- Illumination
- Emission
Materials

- Diffuse
- Normal
- Metalness
- Roughness
Physically Based Rendering

Summary

Physically based shading
SceneKit APIs for materials and lights
Xcode integration
Showcase demo and sample code
HDR Camera and Effects
HDR Camera

HDR is short for High Dynamic Range
Float components
Low dynamic range: 8 bits per components
HDR extends that range
Tone mapped to LDR screens
HDR Camera

Needed for High Dynamic Range contents
Can also be used with normal contents but realistic light ranges

```swift
let camera = SCNCamera()
camera.wantsHDR = true
```
HDR Camera

Tone mapping

Converts from HDR to LDR

Automatic eye adaptation

Configurable (gray point, white point, min/max exposure)

camera.wantsExposureAdaptation = true
camera.averageGray = 0.5
camera.whitePoint = 0.5
camera.exposureOffset = 2.5
camera.minimumExposure = -20.0
camera.maximumExposure = 10.0
HDR Camera
Default exposure
HDR Camera
Under exposure
HDR Camera

Over exposure
Effects

Bloom

High-intensity lights and reflections bleeding on the surrounding pixels

Simulates the effect of being blinded by looking at a bright light

camera.bloomThreshold = 0.5
camera.bloomIntensity = 1.5
camera.bloomBlurRadius = 2.5
Effects

Bloom
Effects

Bloom
Effects

Motion blur

Smoothens camera movements

Some objects can be excluded from the motion blur

```javascript
camera.motionBlurIntensity = 0.2
```
Effects
Motion blur
Effects

Motion blur

Use movability hint to exclude nodes from the motion blur

```python
character.movabilityHint = .movable
```
Effects

Motion blur: Movability hint
Effects

Vignetting

Simulates the round shading aberrations of real camera lenses

camera.vignettingPower = 0.2

camera.vignettingIntensity = 1.2
Effects
Vignetting
Effects

Vignetting
Effects

Color fringe

Simulates the chromatic aberrations happening in real lenses

camera.colorFringeStrength = 0.2
camera.colorFringeIntensity = 0.8
Effects
Color fringe
Effects
Color fringe
Effects

Color correction

Saturation

- Easy black and white look
- Overblown colors

Contrast

- More intense look

camera.saturation = 0.0
camera.contrast = 2.0
Effects
Desaturate
Effects
Saturate
Effects

Color grading

Changes the mood of the rendering by applying a color profile

- 3D color cube
- Lookup table
- Stored as a strip of square images:

```javascript
camera.colorGrading = "colorProfile.png"
```
Effects

Color grading
Effects

Color grading
HDR and Camera effects

Summary

Brand new HDR cameras and effects

- Configurable tone mapping and exposure
- Bloom
- Motion blur
- Vignetting
- Color fringe
- Saturation and contrast
- Color grading
I/O Improvements

Nick Porcino Model I/O Engineer
Primitives
Polygons

Easier to use
Automatic triangulation
Allow for much better subdivision
Opt-in when importing files

let loadingOptions = [.preserveOriginalTopology: true]
Subdivision Surfaces

OpenSubdiv 3

Faster
Better results

Triangles

Quads
Model I/O
Model I/O

3D data interchange

- Between apps
- Between frameworks
- Standard file formats
Model I/O

A new open standard

Years of practical production technologies

Data types specialized for scenes

File layering enables concurrent workflows
Universal Scene Description

Layers
Universal Scene Description

Classes
Universal Scene Description

Variations
Universal Scene Description

Capabilities
Universal Scene Description
Capabilities
Universal Scene Description

Workflow

Integration across the system
Universal Scene Description

Model I/O
Universal Scene Description
Finder

![Finder with USD files](image)
Universal Scene Description

Preview
Universal Scene Description

Xcode
Universal Scene Description

SceneKit
Universal Scene Description

SceneKit
Universal Scene Description

Workflow

Plugins

Seamless movement of 3D data
• Between people
• Content creation programs
• Apps

Plugins and open source information are available from http://openusd.org
Summary

SceneKit available on all platforms
Physically based rendering
HDR camera and effects
Support for USD files
More Information

https://developer.apple.com/wwdc16/609
<table>
<thead>
<tr>
<th>Related Sessions</th>
<th>Location</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual Debugging with Xcode</td>
<td>Presidio</td>
<td>Wednesday 4:00PM</td>
</tr>
<tr>
<td>Working with Wide Color</td>
<td>Mission</td>
<td>Thursday 1:40PM</td>
</tr>
<tr>
<td>Game Technologies for Apple Watch</td>
<td>Mission</td>
<td>Friday 3:00PM</td>
</tr>
<tr>
<td>Lab</td>
<td>Location</td>
<td>Time</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>---------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>SceneKit Lab</td>
<td>Graphics, Games, and Media Lab A</td>
<td>Thursday 3:00PM</td>
</tr>
<tr>
<td>Model I/O Lab</td>
<td>Graphics, Games, and Media Lab B</td>
<td>Thursday 3:00PM</td>
</tr>
<tr>
<td>watchOS Graphics and Games Lab</td>
<td>Graphics, Games, and Media Lab B</td>
<td>Friday 4:00PM</td>
</tr>
</tbody>
</table>