Enhancements to SceneKit

Session 606

Thomas Goossens
Amaury Balliet
Sébastien Métrot
GameKit APIs

High-Level APIs
- SceneKit
- SpriteKit

Services
- Model I/O
- GameplayKit
- GameController

Low-Level APIs
- Metal
- OpenGL
SceneKit

Particles

Physics

Physics Fields

SpriteKit
Scene Editor
Scene Editor
Available in Xcode 7
Scene Editor

Can open and edit DAE, OBJ, Alembic, STL, and PLY files

New native file format (SceneKit archives)
Scene Editor
SceneKit file format

SCNScene archived with NSKeyedArchiver

NSKeyedArchiver.archiveRootObject(scnScene, toFile: aFile)
Scene Editor
Build your game levels
Scene Editor

Particles  
Physics  
Physics Fields  
Actions
Scene Editor

Shader Modifiers

Ambient Occlusion
Demo
Scene Editor

Amaury Balliet
Behind the Scene

Thomas Goossens
Behind the Scene

Concept phase
Behind the Scene
3D modeling
Behind the Scene

Production

- Final models
- Textures
- Lighting
- Skinned character
Behind the Scene
Make it awesome

• Particles
• 2D overlays
• Vegetation
• Fog
Game Sample
Collisions with walls

Rendered Mesh

Collision Mesh
Collisions with the Ground
Collisions with the Ground
Collisions with the Ground
Collisions with the Ground
Collisions with the Ground
Collisions with the Ground
Collisions with the Ground
Collisions with the Ground
Collisions with the Ground
Collisions with the Ground
Animations
Animations
Game Sample

Animated elements
Game Sample
Particles
Game Sample

Particles
Game Sample
Configuring materials
Configuring Materials
Untextured
Configuring Materials

Diffuse texture
Configuring Materials

Normal map
Configuring Materials

Normal map
Configuring Materials
Self-illumination texture
Configuring Materials
Self-illumination texture
Configuring Materials
Reflective cube map
Configuring Materials
Reflective cube map
Configuring Materials

Fresnel
Transition to Metal

Sébastien Métrot
Transition to Metal

Better performance
Modern API
It just works
Transition to Metal
Available with Metal

Compute shaders

- SCNGeometrySource backed by MTLBuffer
- Use MTLTexture as a contents of a material property

Automatic batching
Demo

Metal renderer
Transition to Metal

Adoption

Default on iOS 9
Backward-compatibility ensured
OpenGL can be opted-in
Choosing the Rendering API
Interface Builder
Choosing the Rendering API
Programmatically

let options = [SCNPreferredRenderingAPIKey : SCNRenderingAPI.Metal.rawValue]
aView = SCNView(frame: aFrame, options: options)
Transition to Metal

Dealing with shaders
## Metal Shader Modifiers

<table>
<thead>
<tr>
<th></th>
<th>OpenGL</th>
<th>Metal</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLSL</td>
<td>✔️</td>
<td>✔️ (Translated)</td>
</tr>
<tr>
<td>Metal</td>
<td>❌ (Ignored)</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Metal SCNProgram

No automatic translation

GLSL and Metal

Compiled offline (recommended)

• Only `vertexFunctionName` and `fragmentFunctionName`

Compiled at runtime

• Source code, `vertexFunctionName`, and `fragmentFunctionName`
Metal Shader Declaration

```cpp
#include <metal_stdlib>
using namespace metal;
#include <SceneKit/scn_metal>

struct custom_vertex_t {
    float3 position [[attribute(SCNVertexSemanticPosition)]];
};

struct custom_node_t {
    float4x4 modelViewProjectionTransform;
};

struct MyStruct {
    float3 direction;
    float scale;
};

vertex float4 custom_vert(custom_vertex_t in [[ stage_in ]],
    constant custom_node_t& scn_node [[buffer(1)]],
    constant MyStruct& myArgument [[buffer(0)]])
{
    return float4(in.position * myArgument.scale, 1.0) * scn_node.modelViewProjectionTransform;
}
```
#include <metal_stdlib>
using namespace metal;
#include <SceneKit/scn_metal>

struct custom_vertex_t {
    float3 position [[attribute(SCNVertexSemanticPosition)]];
};

struct custom_node_t {
    float4x4 modelViewProjectionTransform;
};

struct MyStruct {
    float3 direction;
    float scale;
};

vertex float4 custom_vert(custom_vertex_t in [[ stage_in ]],
                            constant custom_node_t& scn_node [[buffer(1)]],
                            constant MyStruct& myArgument [[buffer(0)]])
{
    return float4(in.position * myArgument.scale, 1.0) * scn_node.modelViewProjectionTransform;
}
#include <metal_stdlib>
using namespace metal;
#include <SceneKit/scn_metal>

struct custom_vertex_t {
    float3 position [[attribute(SCNVertexSemanticPosition)]];  
};

struct custom_node_t {
    float4x4 modelViewProjectionTransform;
};

struct MyStruct {
    float3 direction;
    float scale;
};

vertex float4 custom_vert(custom_vertex_t in [[ stage_in ]],
    constant custom_node_t& scn_node [[buffer(1)]],
    constant MyStruct& myArgument [[buffer(0)]])
{
    return float4(in.position * myArgument.scale, 1.0) * scn_node.modelViewProjectionTransform;
}
# Metal Shader Declaration

```cpp
#include <metal_stdlib>
using namespace metal;
#include <SceneKit/scn_metal>

struct custom_vertex_t {
    float3 position [[attribute(SCNVertexSemanticPosition)]];  
};

struct custom_node_t {
    float4x4 modelViewProjectionTransform;
};

struct MyStruct {
    float3 direction;
    float scale;
};

vertex float4 custom_vert(custom_vertex_t in [[ stage_in ]],
    constant custom_node_t& scn_node [[buffer(1)]],
    constant MyStruct& myArgument [[buffer(0)]])
{
    return float4(in.position * myArgument.scale, 1.0) * scn_node.modelViewProjectionTransform;
}
```
#include <metal_stdlib>
using namespace metal;
#include <SceneKit/scn_metal>

struct custom_vertex_t {
    float3 position [[attribute(SCNVertexSemanticPosition)]]; 
};

struct custom_node_t {
    float4x4 modelViewProjectionTransform;
};

struct MyStruct {
    float3 direction;
    float scale;
};

vertex float4 custom_vert(custom_vertex_t in [[ stage_in ]], 
    constant custom_node_t& scn_node [[buffer(1)]],
    constant MyStruct& myArgument [[buffer(0)]])
{
    return float4(in.position * myArgument.scale, 1.0) * scn_node.modelViewProjectionTransform;
}
#include <metal_stdlib>
using namespace metal;
#include <SceneKit/scn_metal>

struct custom_vertex_t {
    float3 position [[attribute(SCNVertexSemanticPosition)]];
};

struct custom_node_t {
    float4x4 modelViewProjectionTransform;
};

struct MyStruct {
    float3 direction;
    float scale;
};

vertex float4 custom_vert(custom_vertex_t in [[ stage_in ]],
    constant custom_node_t& scn_node [[buffer(1)]],
    constant MyStruct& myArgument [[buffer(0)]])
{
    return float4(in.position * myArgument.scale, 1.0) * scn_node.modelViewProjectionTransform;
}
struct MyStruct {
    var direction: float3
    var scale: float
}

let program = SCNProgram()
program.vertexFunctionName = "custom_vert"
program.fragmentFunctionName = "custom_frag"

aMaterial.program = program

var ms = MyStruct( ... );
material.setValue(NSData(bytes:&ms, length:sizeof(MyStruct)),
                 forKey:"myArgument"
struct MyStruct {
    var direction: float3
    var scale: float
}

let program = SCNProgram()
program.vertexFunctionName = "custom_vert"
program.fragmentFunctionName = "custom_frag"

aMaterial.program = program

var ms = MyStruct(
    ...); material.setValue(NSData(bytes:&ms, length:sizeof(MyStruct)), forKey:"myArgument")
struct MyStruct {
    var direction: float3
    var scale: float
}

let program = SCNProgram()
program.vertexFunctionName = "custom_vert"
program.fragmentFunctionName = "custom_frag"

aMaterial.program = program

var ms = MyStruct( ... );
material.setValue(NSData(bytes:&ms, length:sizeof(MyStruct)), forKey:"myArgument")
struct MyStruct {
    var direction: float3
    var scale: float
}

let program = SCNProgram()
program.vertexFunctionName = "custom_vert"
program.fragmentFunctionName = "custom_frag"

aMaterial.program = program

var ms = MyStruct(...);
material.setValue(NSData(bytes:&ms, length:sizeof(MyStruct)), forKey:"myArgument")
Techniques
SCNTechnique

Pass description additions

- `metalVertexShader`
- `metalFragmentShader`
New Features
Integration with Model I/O
## File Formats

<table>
<thead>
<tr>
<th></th>
<th>Geometry</th>
<th>Materials</th>
<th>Animations</th>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAE</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>ABC</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLY</td>
<td>✔️</td>
<td></td>
<td>Not Applicable</td>
<td></td>
</tr>
<tr>
<td>STL</td>
<td>✔️</td>
<td>Not Applicable</td>
<td>Not Applicable</td>
<td>✔️</td>
</tr>
<tr>
<td>OBJ</td>
<td>✔️</td>
<td>✔️</td>
<td>Not Applicable</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Scene Transitions

```swift
aSCNView.presentScene(aScene, withTransition:aSKTransition, incomingPointOfView:nil, completionHandler:nil)
```
Scene Transitions

```swift
a SCNView.presentScene(aScene, withTransition:a SKTransition, incomingPointOfView:nil, completionHandler:nil)
```
Debug Options

aSCNView.debugOptions = .ShowBoundingBoxes | .ShowPhysicsShapes
Blend Modes

cSCNMaterial.blendMode = .Add
Audio Nodes

Sounds move with nodes in 3D
Ambience and music
Listener on the camera by default
AVAudioNode extensible
SCNAction to play sound
Audio Nodes

3D sounds

```swift
let source = SCNAudioSource(named: "sound.caf")
let player = SCNAudioPlayer(source: source)
in node.addAudioPlayer(player);
```

Ambience and music

```swift
source.positional = false
source.loops = true
```

Play with SCNAction

```swift
let action = SCNAction.playAudioSource(source, waitForCompletion: true)
in node.runAction(action)
```
Audio Nodes

Game sample

Play the right step sound based on the type of ground
Audio Nodes

Game sample

Play the right step sound based on the type of ground
Audio Nodes

Game sample

Play the right step sound based on the type of ground
Enhancements to SceneKit
More Information

SceneKit Documentation and Videos
http://developer.apple.com/scenekit

Apple Developer Forums
http://developer.apple.com/forums

Developer Technical Support
http://developer.apple.com/support/technical

General Inquiries
Allan Schaffer, Game Technologies Evangelist
aschaffer@apple.com
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Type</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managing 3D Assets with Model I/O</td>
<td>Mission</td>
<td>Tuesday 2:30PM</td>
</tr>
<tr>
<td>What’s New in SpriteKit</td>
<td>Mission</td>
<td>Wednesday 10:00AM</td>
</tr>
<tr>
<td>Labs</td>
<td>Graphics, Games, and Media Lab C</td>
<td>Wednesday 3:30PM</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>SceneKit Lab</td>
<td>Graphics, Games, and Media Lab B</td>
<td>Thursday 2:30PM</td>
</tr>
</tbody>
</table>