Scene Kit

Session 504

Thomas Goossens
Software engineer

Scene Kit

- New framework on Mountain Lion
- Eases the integration of 3D into applications

Common 3D Use Cases

- 3D user interfaces
- Showcases, presentation
- Data visualization
- Games

Common 3D Use Cases

- 3D user interfaces
- Showcases, presentation
- Data visualization
- Games

Common 3D Use Cases

- 3D user interfaces
- Showcases, presentation
- Data visualization
- Games

Common 3D Use Cases

- 3D user interfaces
- Showcases, presentation
- Data visualization
- Games

Common 3D Use Cases

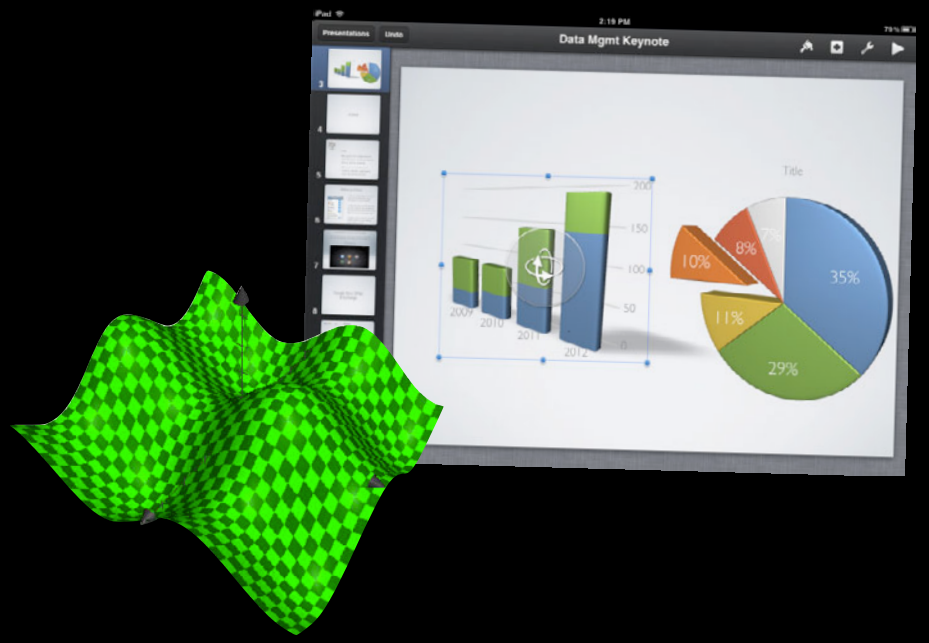
- 3D user interfaces
- Showcases, presentation
- Data visualization
- Games

Common 3D Use Cases

- 3D user interfaces
- Showcases, presentation
- Data visualization
- Games

Common 3D Use Cases

- 3D user interfaces
- Showcases, presentation
- Data visualization
- Games



Common 3D Use Cases

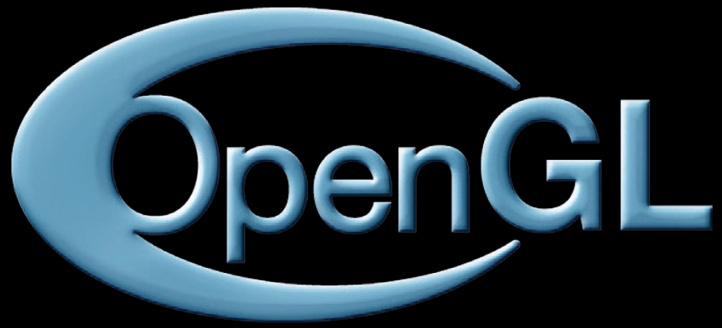
- 3D user interfaces
- Showcases, presentation
- Data visualization
- Games

Common 3D Use Cases

- 3D user interfaces
- Showcases, presentation
- Data visualization
- Games

3D on OS X

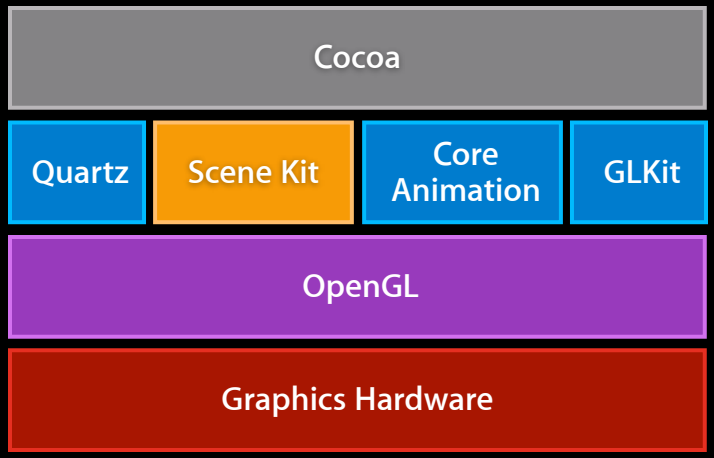
- High-performance rendering APIs
  - OpenGL
    - GLSL
    - GLKit
- Low-level APIs
- Require advanced skills



“Scene Kit is a high-level
API on top of OpenGL that
operates on a scene graph”

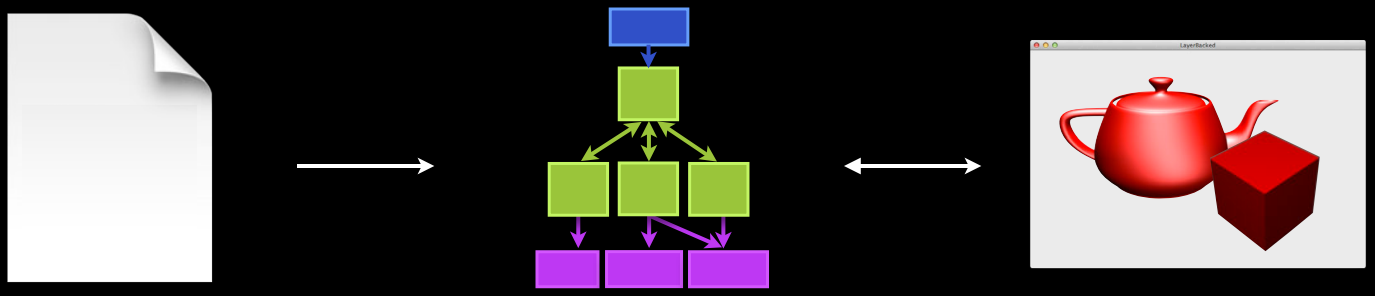
Scene Kit

- High-level Objective-C API
- Great flexibility
- Integrated with Cocoa and Core Animation



Scene Kit

Load / manipulate / render



3D File Format | Scene Graph | Rendering

Loading a 3D Scene

- It's easier to create complex scenes using 3D tools
- Minimize the code
- Work with artists

Loading a 3D Scene

Digital Asset Exchange Documents

- XML based
- Supported by the major 3D tools
- Popular in the industry

3D Assets Loading

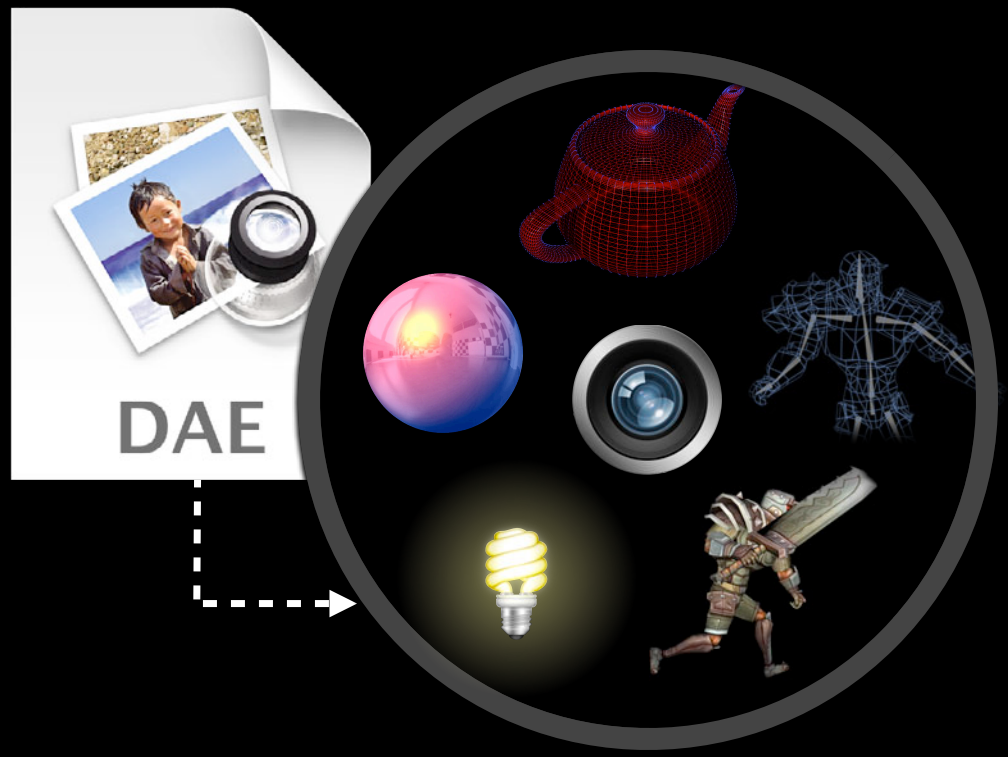
DAE documents

- Geometry
- Animations
- Materials
- Lighting
- Point of views
- Skinning
- Morphing

3D Assets Loading

DAE documents

- Geometry
- Animations
- Materials
- Lighting
- Point of views
- Skinning
- Morphing



Render a 3D Scene

Scene creation code example

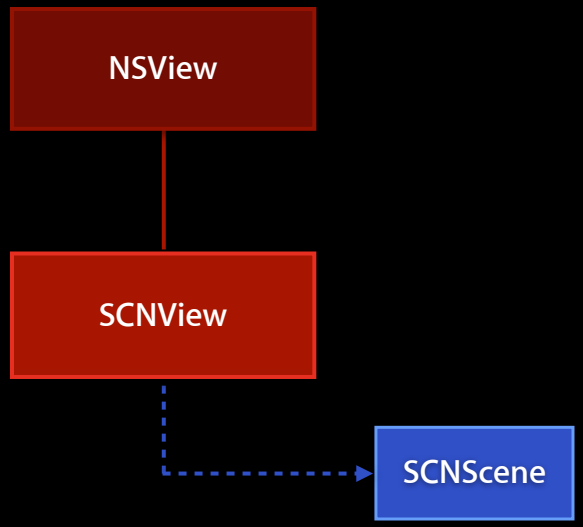
```
NSURL *url = [[NSBundle mainBundle] URLForResource:@"myScene"
                                      withExtension:@"dae"];

NSError *error;
SCNScene *scene = [SCNScene sceneWithURL:url
                                 options:nil
                                   error:&error];
```

Rendering a Scene

View creation code example

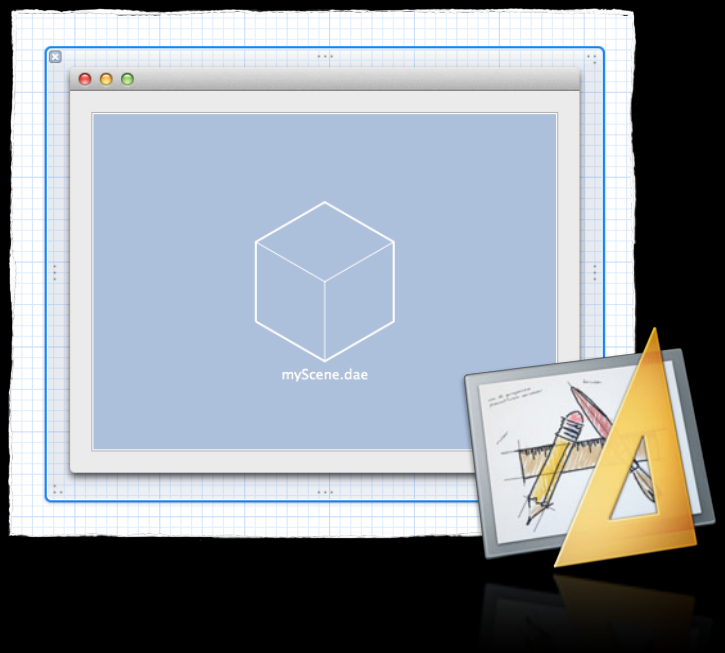
```
SCNView *view = [[SCNView alloc] initWithFrame:frame options:nil];
view.scene = scene;
```



Rendering a Scene

View creation in Interface Builder

- Drag an SCNView from the library

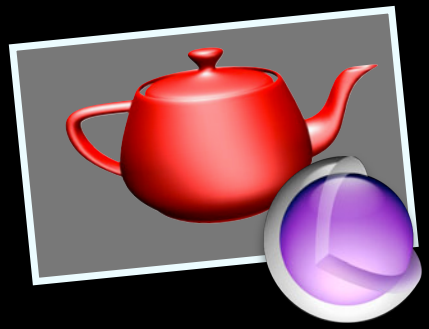


Demo

Rendering a Scene

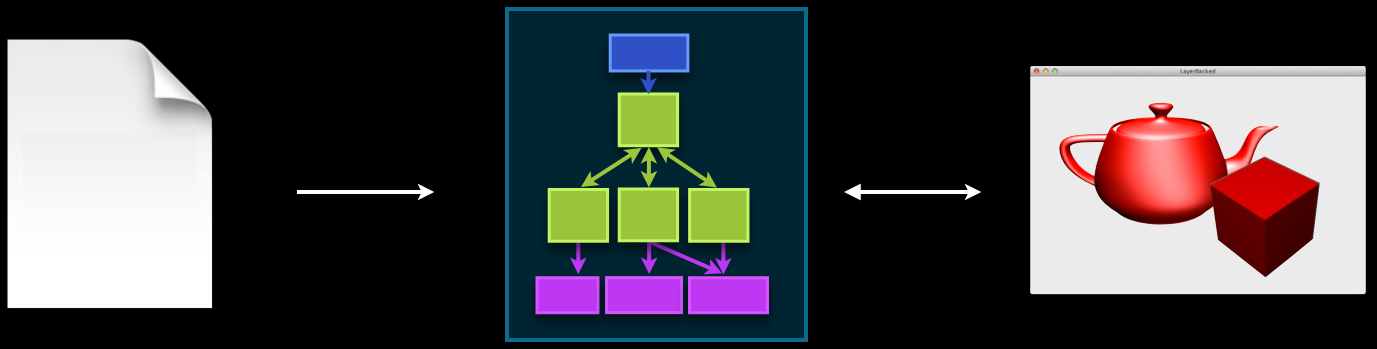
SCNView

SCNLayer



SCNRenderer

Manipulating a Scene



3D File Format | Scene Graph | Rendering

Manipulating a Scene

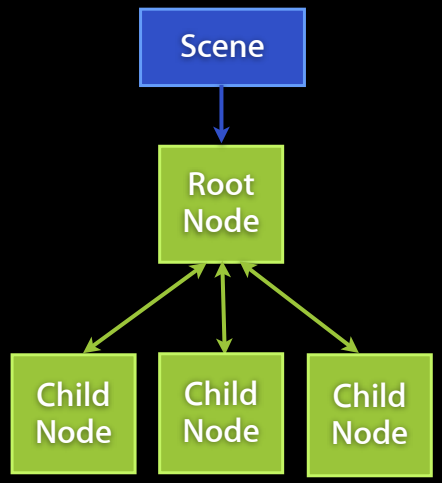
Examples

- Move, scale, and rotate elements
- Animates
- Change colors and images
- Change the lighting
- And more

Scene Manipulation

Structure

- Node tree
  - Similar to view hierarchy and layer tree
  - SCNNode
- A node is a coordinate system in 3D space
  - Relative to its parent node
  - Position
  - Rotation
  - Scale



Scene Manipulation

Manipulation code example

```
// Move a node to another position.
myNode.position = SCNVector3Make(0, 0, 0);

// Add a child node.
[myNode addChildNode:anotherNode];

// Remove a node from its parent node.
[anotherNode removeFromParentNode];
```

Scene Manipulation

Manipulation code example

```
// Move a node to another position.
myNode.position = SCNVector3Make(0, 0, 0);
```

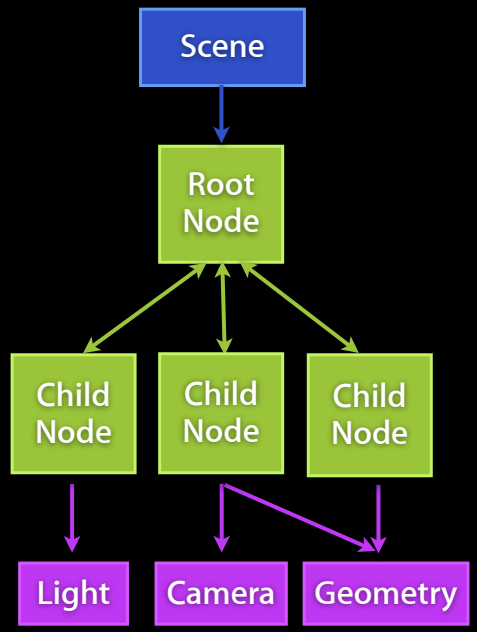
```
// Add a child node.
[myNode addChildNode:anotherNode];

// Remove a node from its parent node.
[anotherNode removeFromParentNode];
```

Scene Manipulation

Node attributes

- Camera
- Light
- Geometry
- Can be shared



Node Attributes

Camera

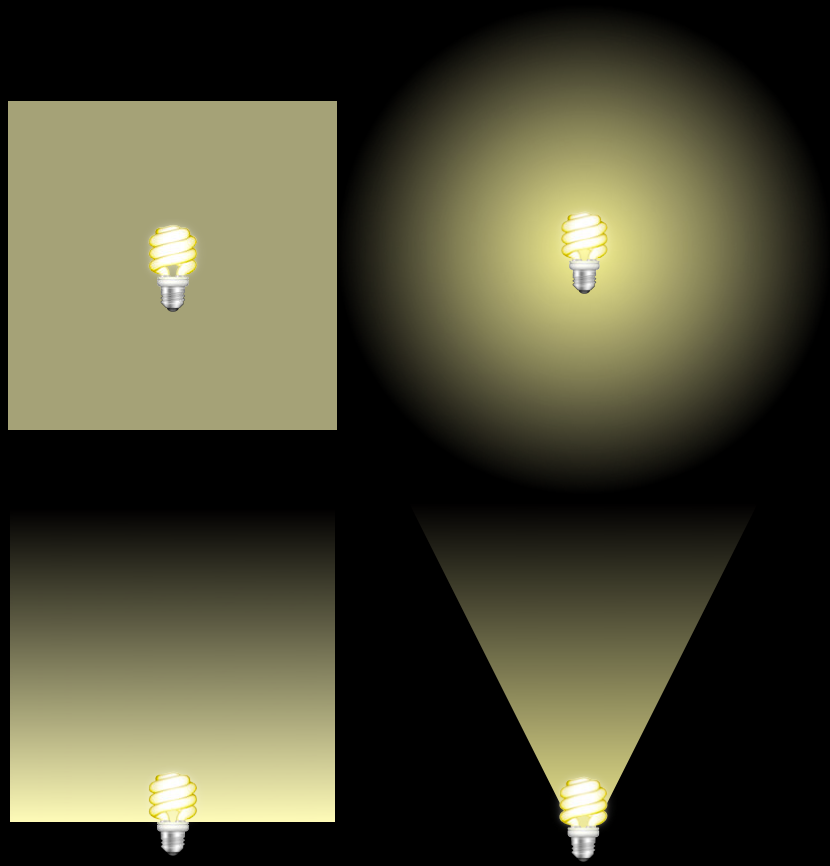
- Point of view for renderers

```
sceneView.pointOfView = topCameraNode;  //set a new point of view
topCameraNode.camera.yFov = 60.0;       //tweak a parameter of the camera
```

Node Attributes

Light

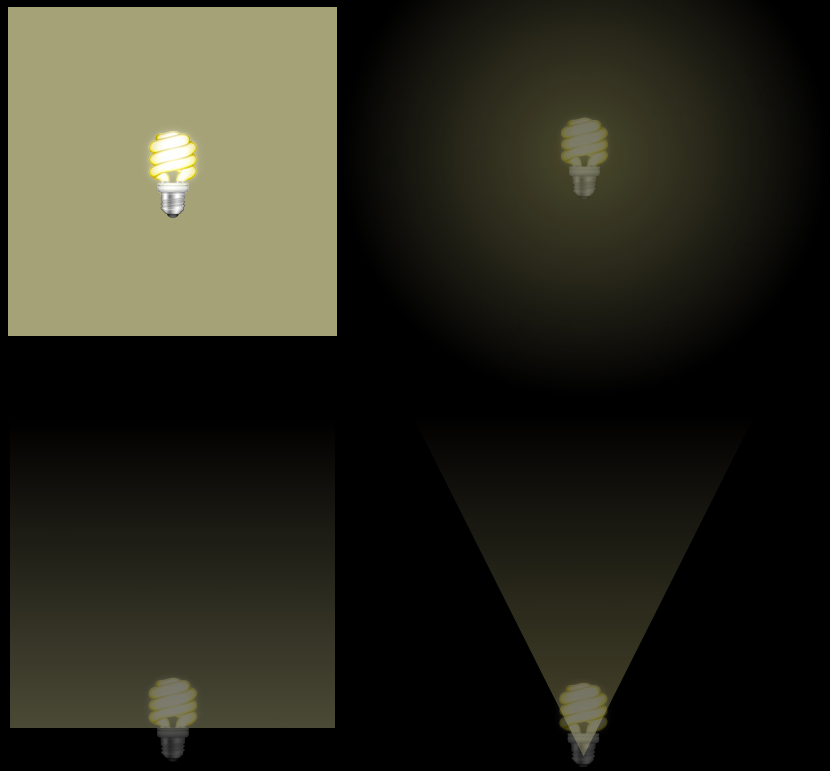
- Ambient
- Omni
- Directional
- Spot



Node Attributes

Light

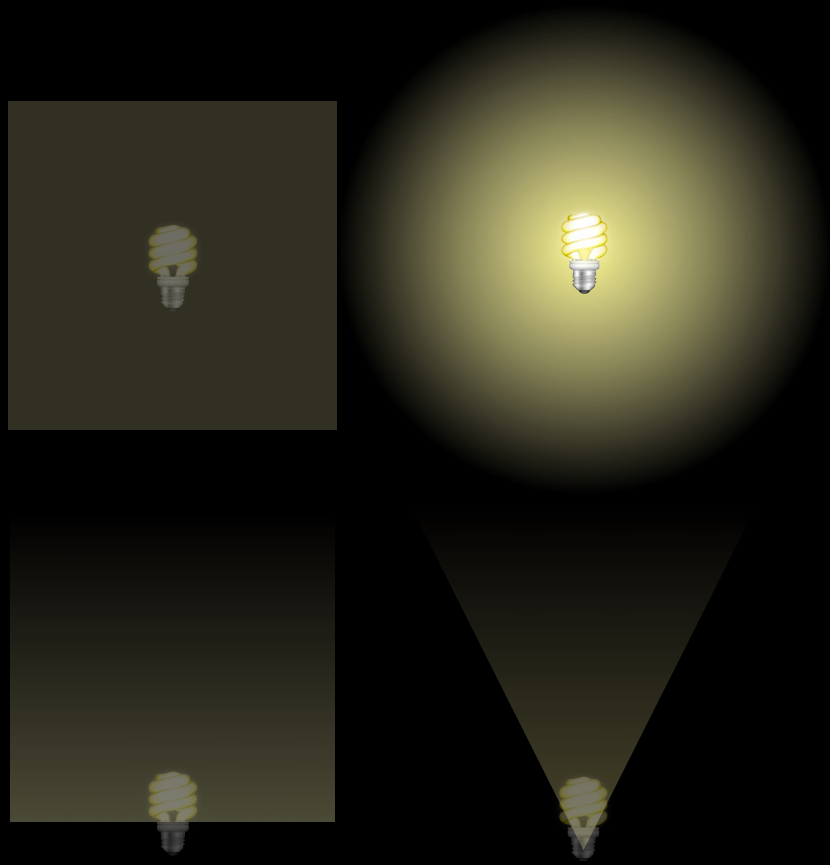
- Ambient
- Omni
- Directional
- Spot



Node Attributes

Light

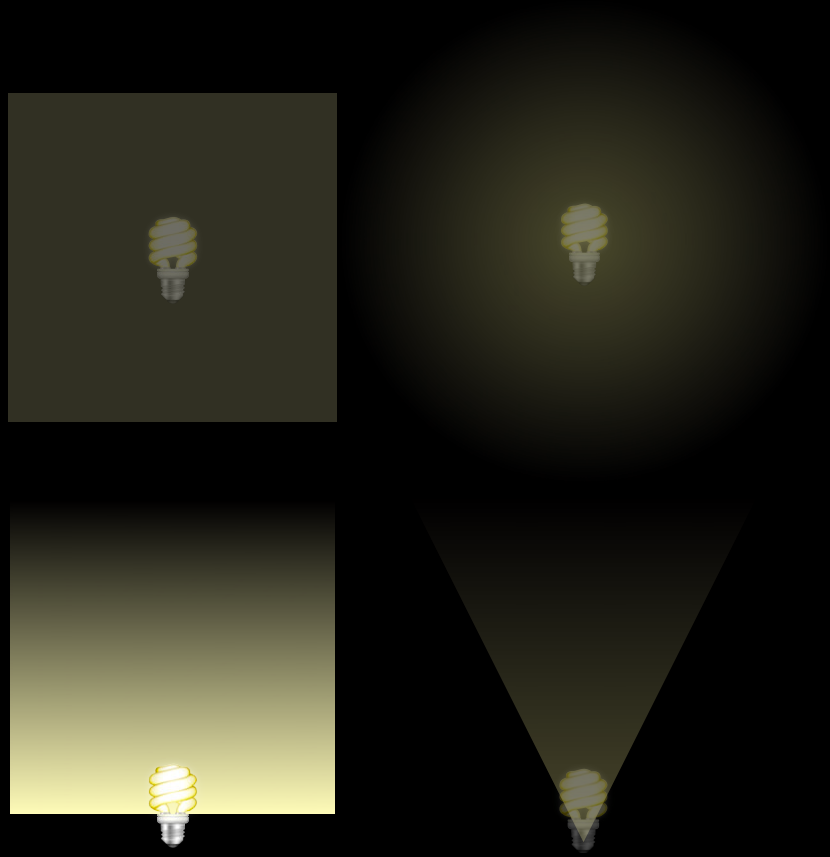
- Ambient
- Omni
- Directional
- Spot



Node Attributes

Light

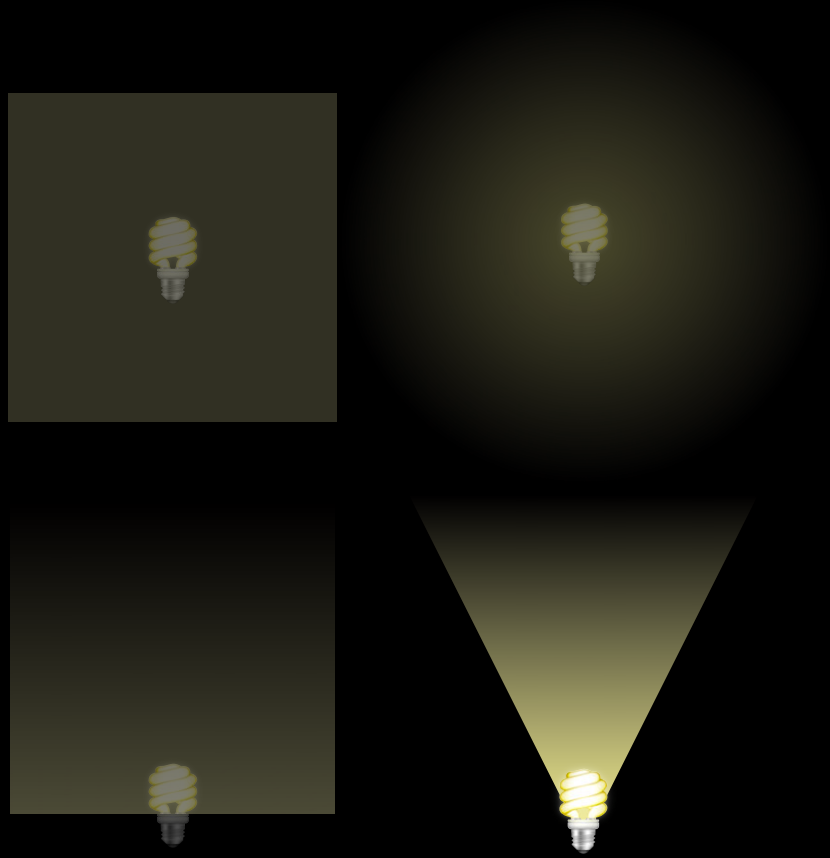
- Ambient
- Omni
- Directional
- Spot



Node Attributes

Light

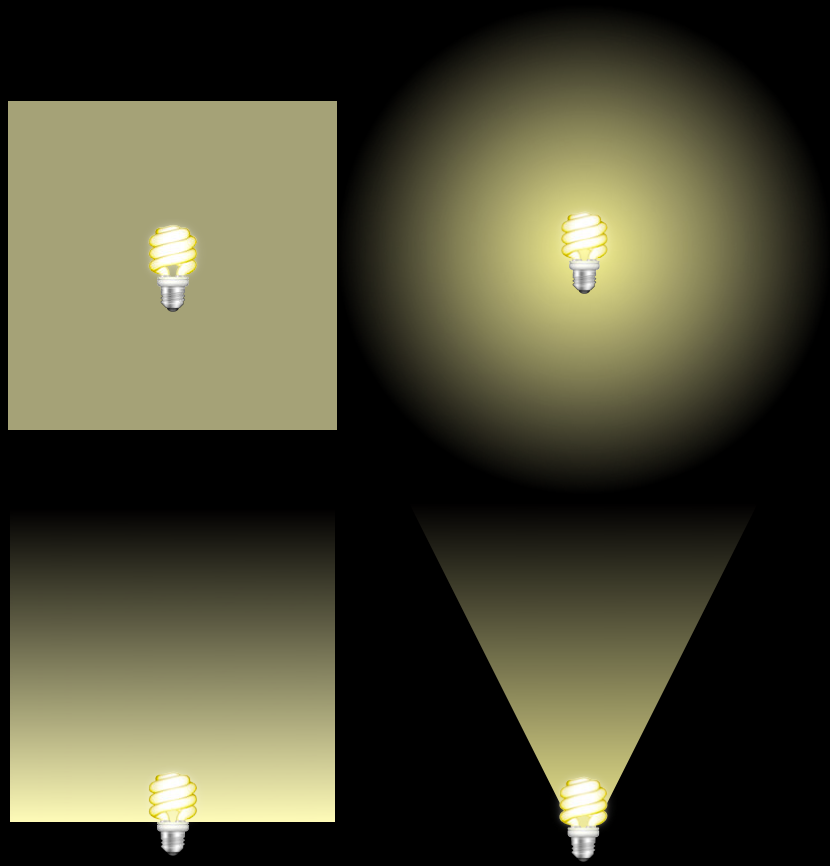
- Ambient
- Omni
- Directional
- Spot



Node Attributes

Light

- Ambient
- Omni
- Directional
- Spot



Node Attributes

Geometry

- Triangles
- Vertices
- Normals
- UVs
- Materials

code example

node.geometry

change a material

Node Attributes

Geometry

- Triangles
- Vertices
- Normals
- UVs
- Materials

code example

node.geometry

change a material

Node Attributes

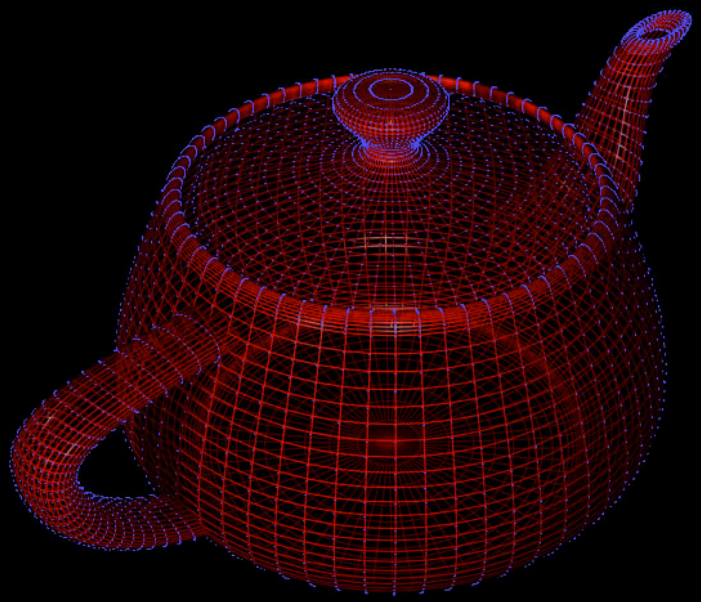
Geometry

- Triangles
- Vertices
- Normals
- UVs
- Materials

code example

node.geometry

change a material



Node Attributes

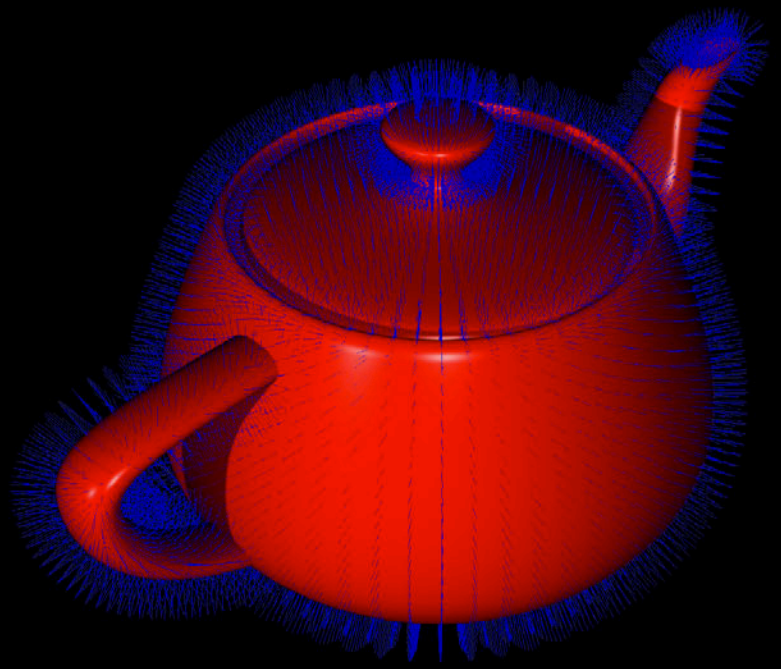
Geometry

- Triangles
- Vertices
- Normals
- UVs
- Materials

code example

node.geometry

change a material



Node Attributes

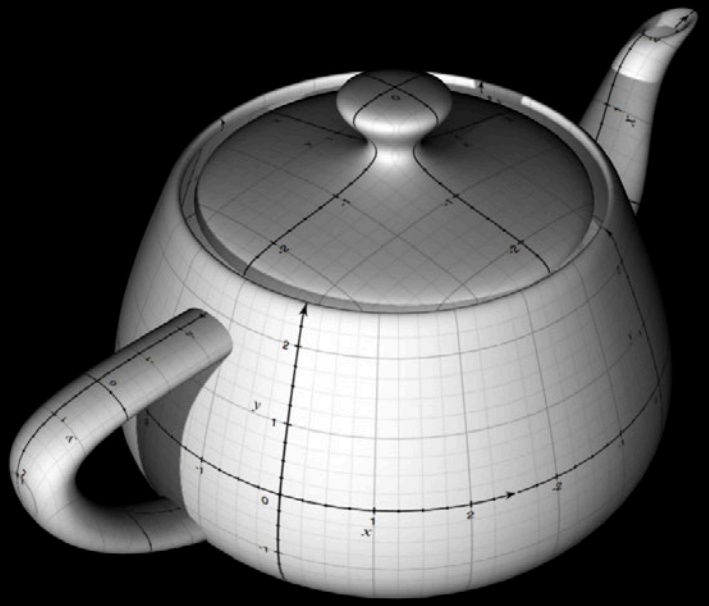
Geometry

- Triangles
- Vertices
- Normals
- UVs
- Materials

code example

node.geometry

change a material



Node Attributes

Geometry

- Triangles
- Vertices
- Normals
- UVs
- Materials

code example

node.geometry

change a material

Node Attributes

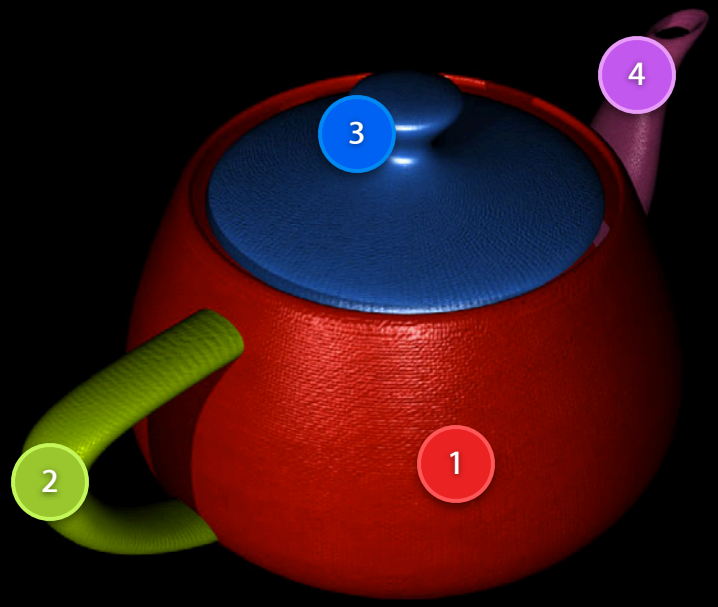
Geometry

- Triangles
- Vertices
- Normals
- UVs
- Materials

code example

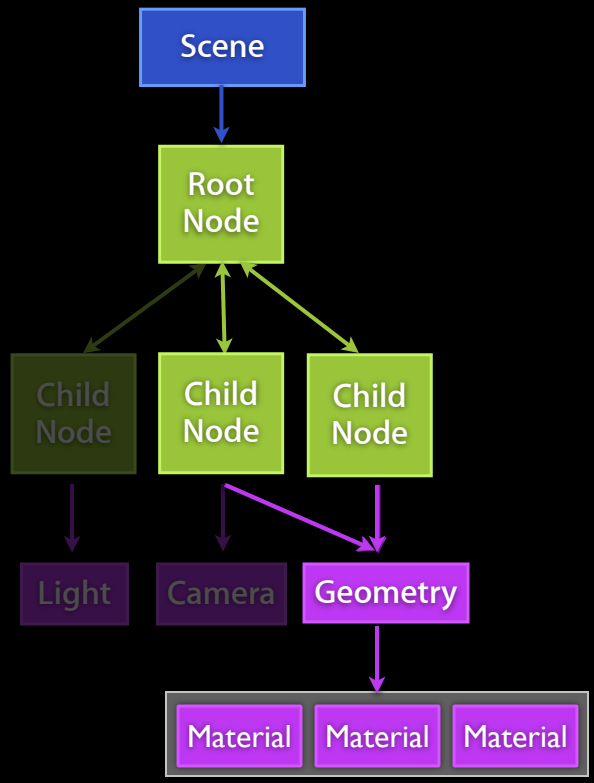
node.geometry

change a material



Materials

- Determine the geometry appearance
  - SCNMaterial
  - May depend on lights
- Material properties
  - SCNMaterialProperty
  - Contents is a color or an image
  - Eight properties



Material Properties

- Diffuse
- Ambient
- Specular and shininess
- Reflective
- Transparent
- Normal
- Multiply
- Emission

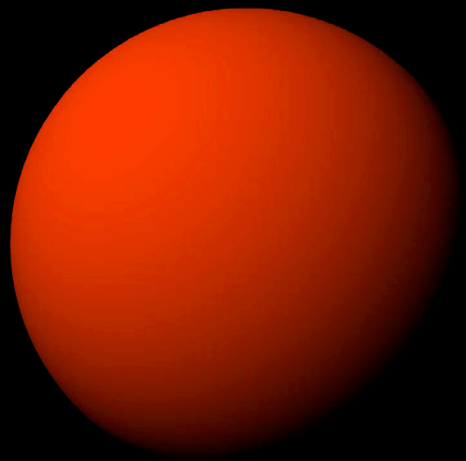
Material Properties

- Diffuse
- Ambient
- Specular and shininess
- Reflective
- Transparent
- Normal
- Multiply
- Emission

```
material.diffuse.contents = anImage;
```

Material Properties

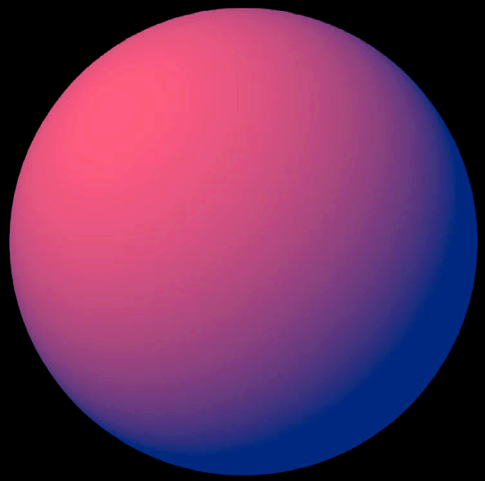
- Diffuse
- Ambient
- Specular and shininess
- Reflective
- Transparent
- Normal
- Multiply
- Emission



```
material.diffuse.contents = [NSColor redColor];
```

Material Properties

- Diffuse
- Ambient
- Specular and shininess
- Reflective
- Transparent
- Normal
- Multiply
- Emission



```
material.ambient.contents = [NSColor blueColor];
```

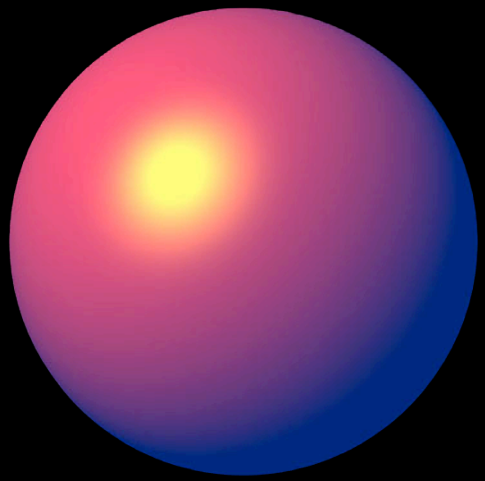
Material Properties

- Diffuse
- Ambient
- Specular and shininess
- Reflective
- Transparent
- Normal
- Multiply
- Emission

```
material.specular.contents = [NSColor yellowColor];
material.shininess = 0.1;
```

Material Properties

- Diffuse
- Ambient
- Specular and shininess
- Reflective
- Transparent
- Normal
- Multiply
- Emission



```
material.specular.contents = [NSColor yellowColor];
material.shininess = 0.9;
```

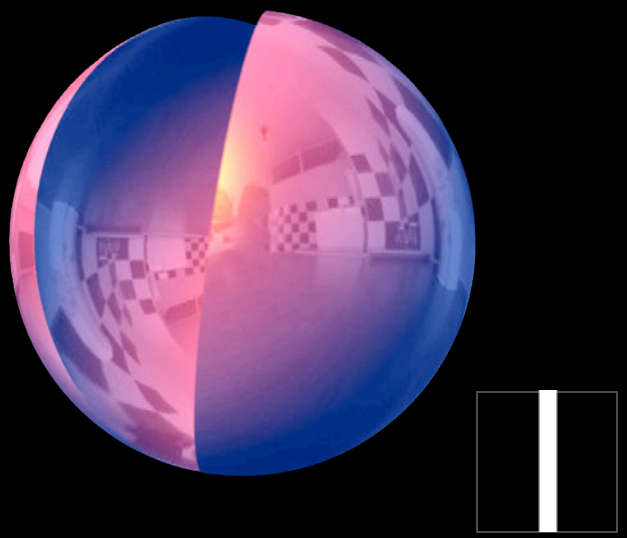
Material Properties

- Diffuse
- Ambient
- Specular and shininess
- Reflective
- Transparent
- Normal
- Multiply
- Emission

```
material.reflective.contents = anImage;
```

Material Properties

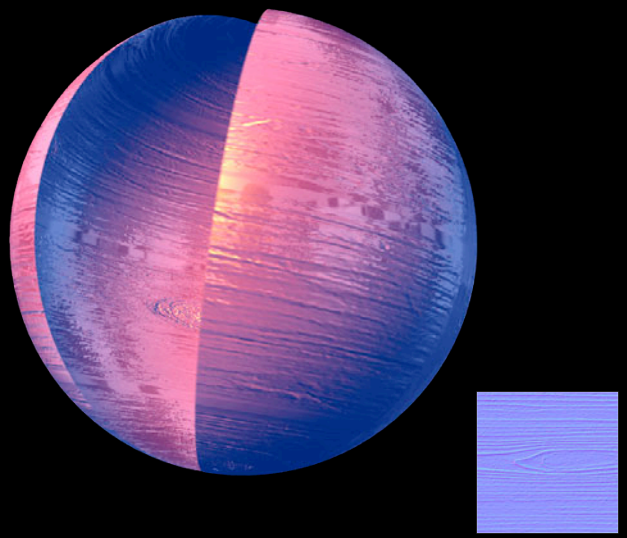
- Diffuse
- Ambient
- Specular and shininess
- Reflective
- Transparent
- Normal
- Multiply
- Emission



```
material.transparent.contents = anImage;
```

Material Properties

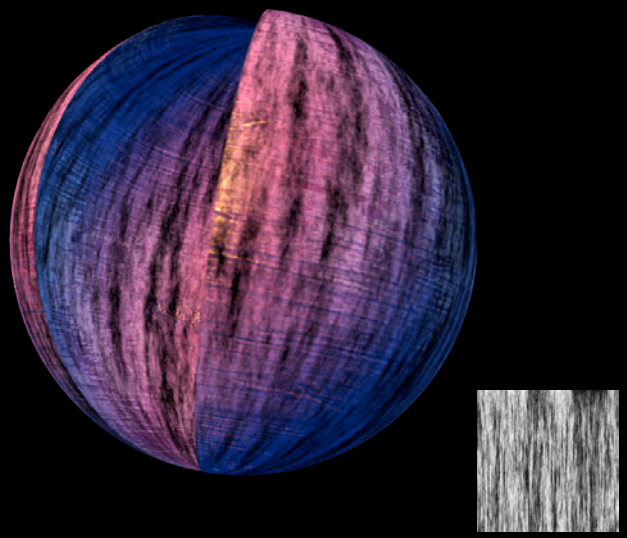
- Diffuse
- Ambient
- Specular and shininess
- Reflective
- Transparent
- Normal
- Multiply
- Emission



```
material.normal.contents = anImage;
```

Material Properties

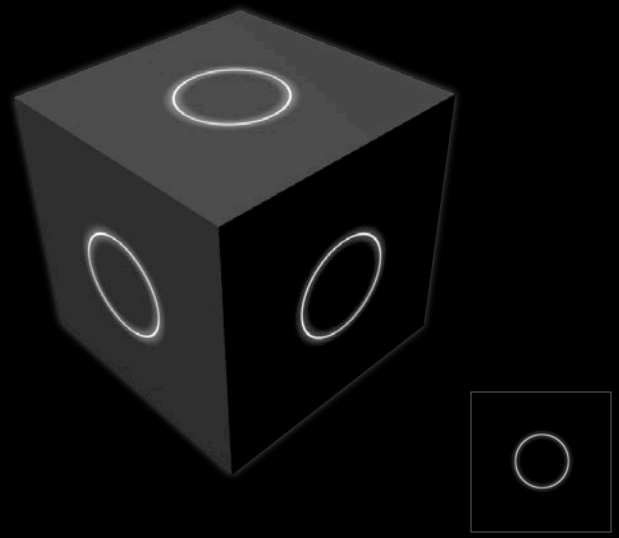
- Diffuse
- Ambient
- Specular and shininess
- Reflective
- Transparent
- Normal
- Multiply
- Emission



```
material.multiply.contents = anImage;
```

Material Properties

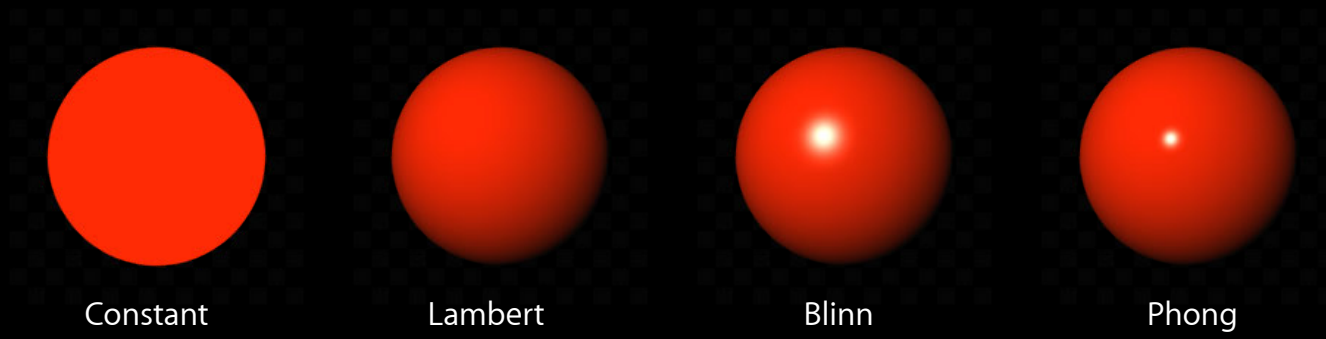
- Diffuse
- Ambient
- Specular and shininess
- Reflective
- Transparent
- Normal
- Multiply
- Emission



```
material.emission.contents = anImage;
```

Materials

Lighting models




```
material.lightingModel = SCNLightingModelBlinn;
```

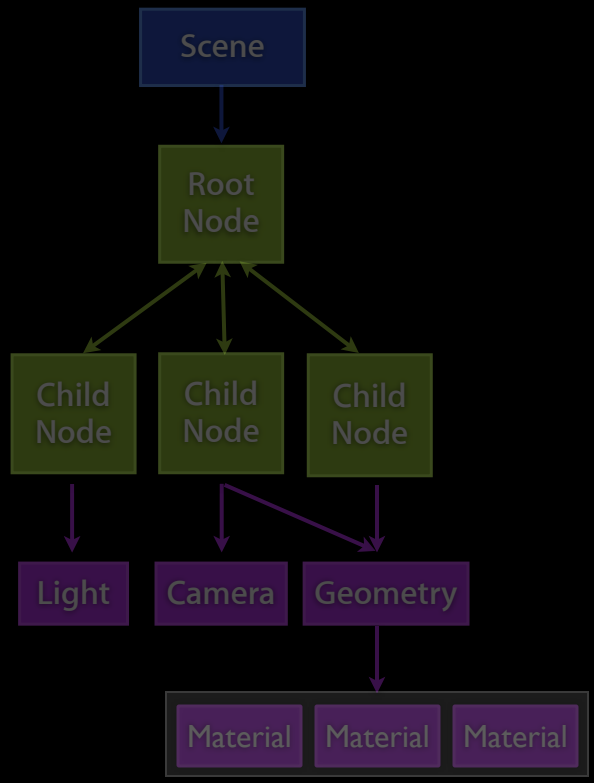
Materials

Configure and set a material example

```
// Access the geometry attribute of a node.
SCNGeometry *geometry = node.geometry;


// Create a new “red” material.
SCNMaterial *aMaterial = [SCNMaterial material];
aMaterial.diffuse.contents = [NSColor redColor];


// Set this material to our geometry
geometry.firstMaterial = aMaterial;
```



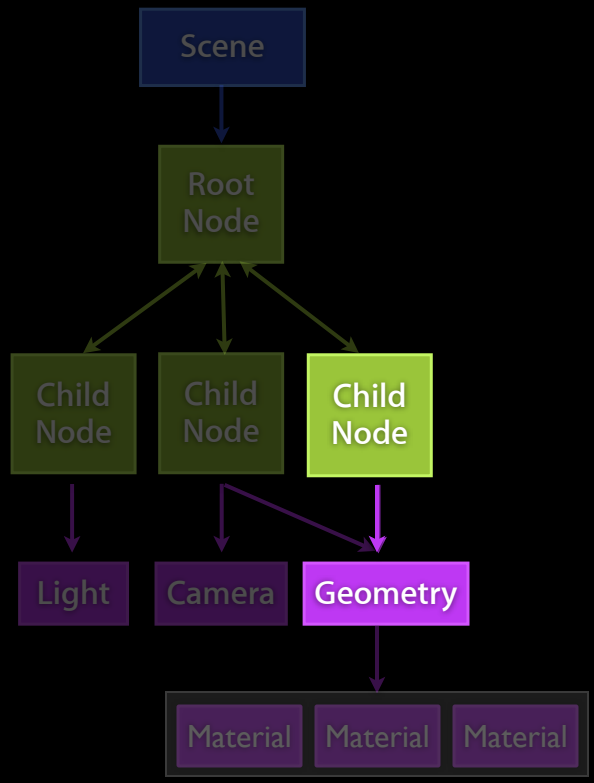
Materials

Configure and set a material example

```
// Access the geometry attribute of a node.
SCNGeometry *geometry = node.geometry;
```

```
// Create a new “red” material.
SCNMaterial *aMaterial = [SCNMaterial material];
aMaterial.diffuse.contents = [NSColor redColor];
```

```
// Set this material to our geometry
geometry.firstMaterial = aMaterial;
```



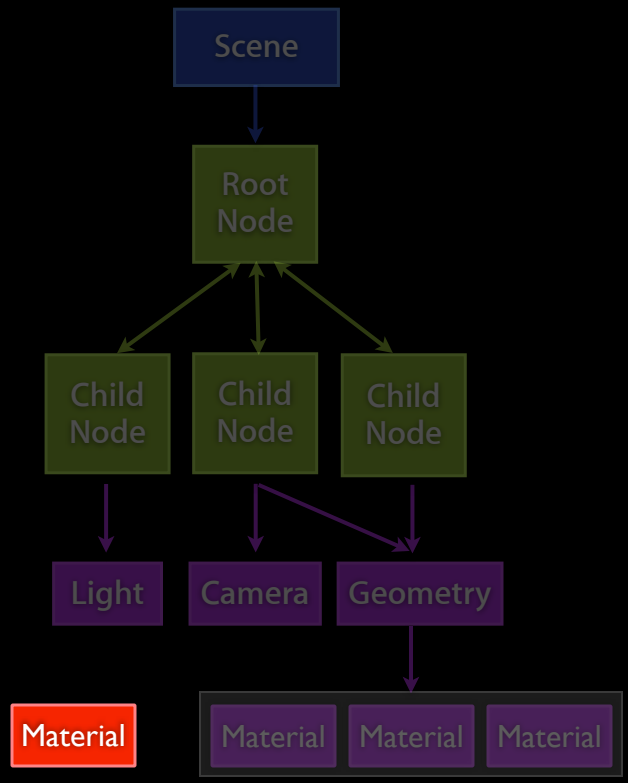
Materials

Configure and set a material example

```
// Access the geometry attribute of a node.
SCNGeometry *geometry = node.geometry;
```

```
// Create a new “red” material.
SCNMaterial *aMaterial = [SCNMaterial material];
aMaterial.diffuse.contents = [NSColor redColor];
```

```
// Set this material to our geometry
geometry.firstMaterial = aMaterial;
```



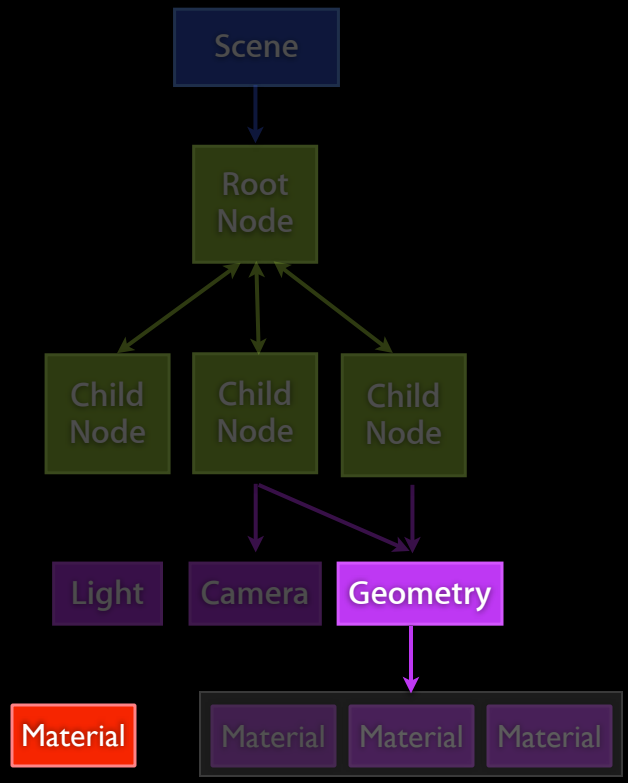
Materials

Configure and set a material example

```
// Access the geometry attribute of a node.
SCNGeometry *geometry = node.geometry;


// Create a new “red” material.
SCNMaterial *aMaterial = [SCNMaterial material];
aMaterial.diffuse.contents = [NSColor redColor];
```

```
// Set this material to our geometry
geometry.firstMaterial = aMaterial;
```



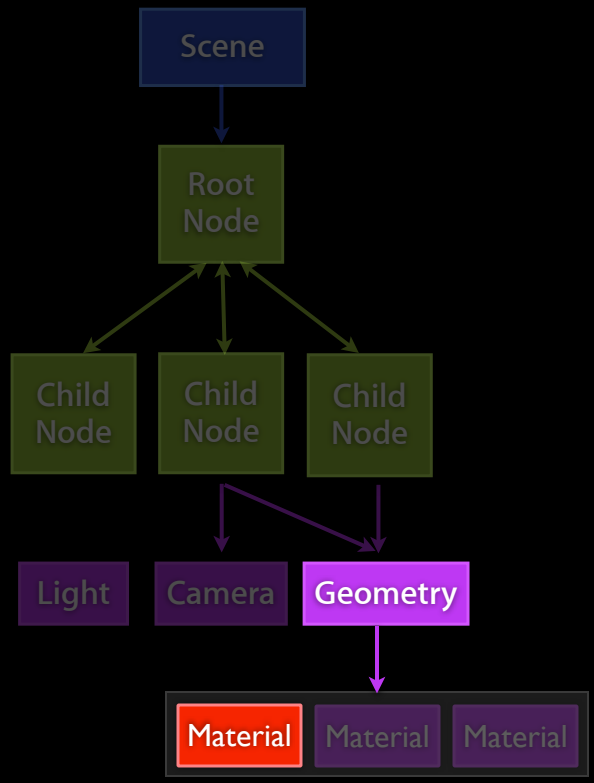
Materials

Configure and set a material example

```
// Access the geometry attribute of a node.
SCNGeometry *geometry = node.geometry;


// Create a new “red” material.
SCNMaterial *aMaterial = [SCNMaterial material];
aMaterial.diffuse.contents = [NSColor redColor];
```

```
// Set this material to our geometry
geometry.firstMaterial = aMaterial;
```



Building an Application with Scene Kit

Amaury Balliet
Software engineer

3D Assets on the Mac

DAE files

3D Assets on the Mac

DAE files

Preview

Quick Look

3D Assets on the Mac

DAE files

Preview

Quick Look

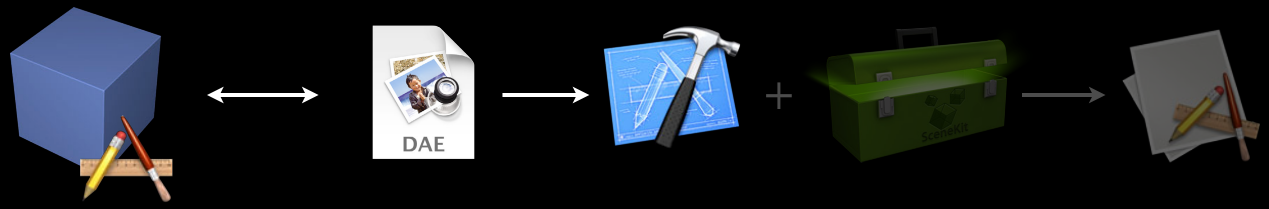
Preview Inspect Adjust

Typical Workflow

- Built into Xcode
- Scene graph inspection
- Node attributes adjustment
- Preview animations and performance

Typical Workflow

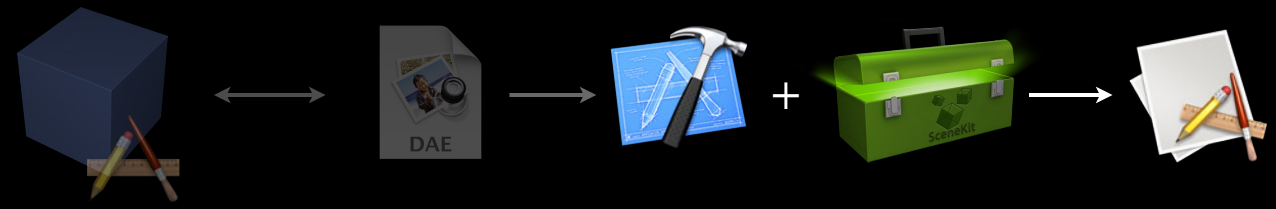
Artists



Authoring Tools | 3D format | Integration | Application

Typical Workflow

Engineers



Authoring Tools

3D format

Integration

Application

Demo

Building an App with Scene Kit

Adjusting a scene

- Retrieve nodes by name

```
SCNNode *node = [scene.rootNode childNodeWithName:@"nameOfMyNode"
                                        recursively:YES];
```

Building an App with Scene Kit

Adjusting a scene

- Retrieve nodes by name

```
SCNNode *node = [scene.rootNode childNodeWithName:@"nameOfMyNode"
                                        recursively:YES];
```

- Retrieve materials

```
SCNMaterial *material = node.geometry.firstMaterial;
```

Building an App with Scene Kit

Adjusting a scene

- Retrieve nodes by name

```
SCNNode *node = [scene.rootNode childNodeWithName:@"nameOfMyNode"
                                        recursively:YES];
```

- Retrieve materials

```
SCNMaterial *material = node.geometry.firstMaterial;
```

- Customize materials

```
material.diffuse.contents = [NSImage imageNamed:@"texture"];
```

Going Beyond

Aymeric Bard
Software engineer

Going Beyond

Agenda

- Animations
- Extending Scene Kit with OpenGL
- Creating geometries
- Mixing Core Animation with Scene Kit

Animations

- Almost everything is animatable
- Implicit animations
- Explicit animations
- Same programming model as Core Animation

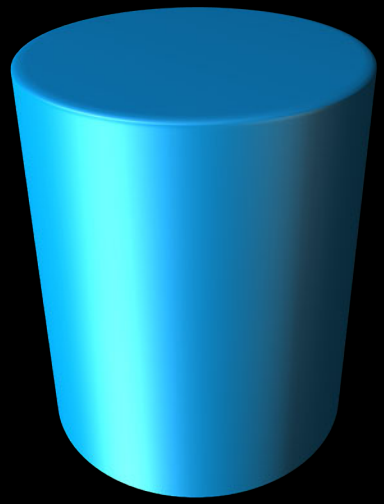
Animations

Implicit animations

```
// Begin a transaction.
[SCNTransaction begin];
[SCNTransaction setAnimationDuration:2.0];


// Change a property.
aNode.opacity = 0.2;


// Commit the transaction.
[SCNTransaction commit];
```



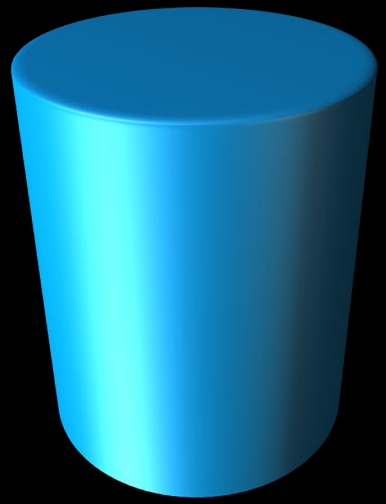
Animations

Implicit animations

```
// Begin a transaction.
[SCNTransaction begin];
[SCNTransaction setAnimationDuration:2.0];


// Change a property.
aNode.opacity = 0.2;


// Commit the transaction.
[SCNTransaction commit];
```



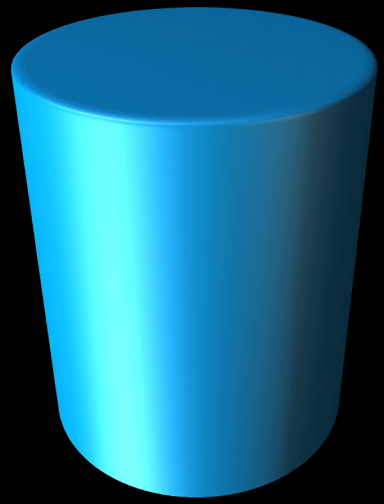
Animations

Implicit animations

```
// Begin a transaction.
[SCNTransaction begin];
[SCNTransaction setAnimationDuration:2.0];

// Change a property.
aNode.opacity = 0.2;

// Commit the transaction.
[SCNTransaction commit];
```



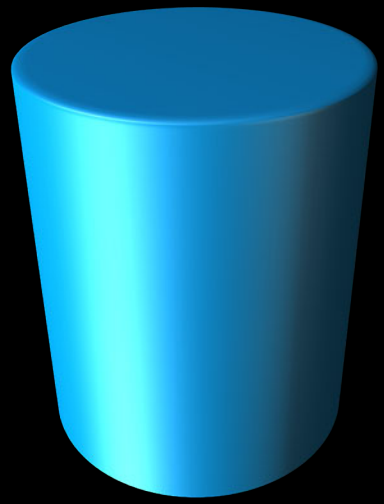
Animations

Implicit animations

```
// Begin a transaction.
[SCNTransaction begin];
[SCNTransaction setAnimationDuration:2.0];


// Change a property.
aNode.opacity = 0.2;


// Commit the transaction.
[SCNTransaction commit];
```



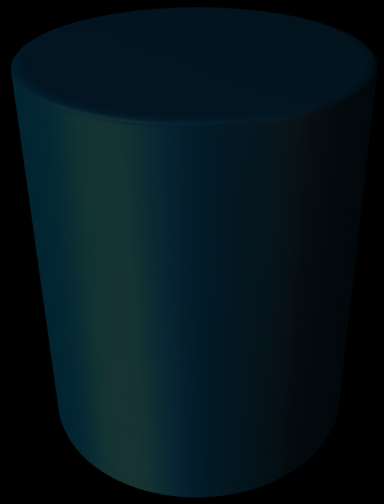
Animations

Implicit animations

```
// Begin a transaction.
[SCNTransaction begin];
[SCNTransaction setAnimationDuration:2.0];


// Change a property.
aNode.opacity = 0.2;


// Commit the transaction.
[SCNTransaction commit];
```



Animations

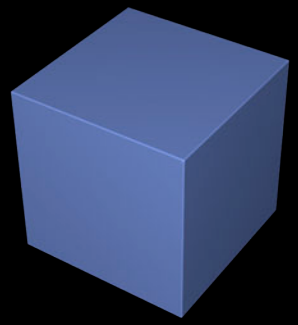
Explicit animations

- CABasicAnimation
- CAKeyframeAnimation
- CAAnimationGroup

Animations

Explicit animations

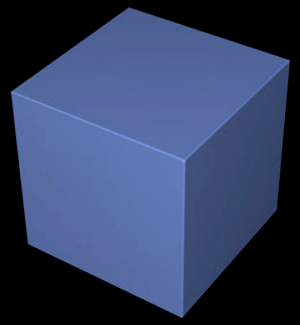
- CABasicAnimation
- CAKeyframeAnimation
- CAAnimationGroup



Animations

Explicit animations

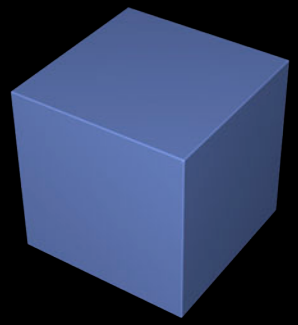
- CABasicAnimation
- CAKeyframeAnimation
- CAAnimationGroup



Animations

Explicit animations

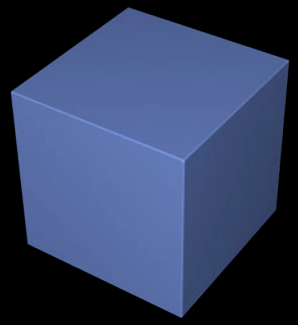
- CABasicAnimation
- CAKeyframeAnimation
- CAAnimationGroup



Animations

Explicit animations

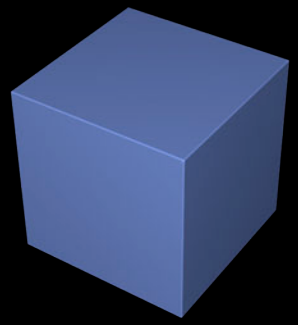
- CABasicAnimation
- CAKeyframeAnimation
- CAAnimationGroup



Animations

Explicit animations

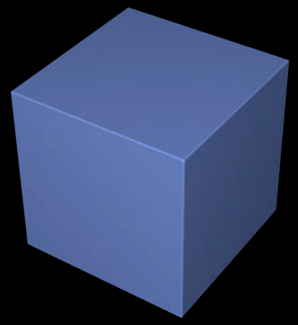
- CABasicAnimation
- CAKeyframeAnimation
- CAAnimationGroup



Animations

Explicit animations

- CABasicAnimation
- CAKeyframeAnimation
- CAAnimationGroup



Animations

Explicit animations

```
// Create an animation.
animation = [CABasicAnimation animationWithKeyPath:@"opacity"];

// Configure the animation.
animation.duration = 2.0;
animation.toValue = [NSNumber numberWithFloat:0.2];

// Play the animation.
[aNode addAnimation:animation forKey:@"myOpacityAnimation"];
```

Animations

Explicit animations

```
// Create an animation.
animation = [CABasicAnimation animationWithKeyPath:@"opacity"];

// Configure the animation.
animation.duration = 2.0;
animation.toValue = [NSNumber numberWithFloat:0.2];

// Play the animation.
[aNode addAnimation:animation forKey:@"myOpacityAnimation"];
```

Animations

Explicit animations

```
// Create an animation.
animation = [CABasicAnimation animationWithKeyPath:@"opacity"];


// Configure the animation.
animation.duration = 2.0;
animation.toValue = [NSNumber numberWithFloat:0.2];

// Play the animation.
[aNode addAnimation:animation forKey:@"myOpacityAnimation"];
```

Mixing OpenGL and Scene Kit

- Scene delegate rendering
- Node delegate rendering
- Material custom program

Extending Scene Kit with OpenGL

Scene delegate rendering

- Custom GL code, free of constraints
- Before and/or after scene rendering
- Usable on SCNView, SCNLayer, and SCNRenderer

Extending Scene Kit with OpenGL

Scene delegate rendering

Mixing OpenGL and Scene Kit

Scene delegate rendering code example

```
- (void)renderer:(id <SCNSceneRenderer>)aRenderer
 willRenderScene:(SCNScene *)scene
          atTime:(NSTimeInterval)time
{
    // Custom OpenGL code
    glBindVertexArrayAPPLE(myVAO);
    glDisable(GL_DEPTH_TEST);
    glUseProgram(myProgram);
    CGSize size = self.myView.frame.size;
    glUniform2f(myResolutionLoc, size.width, size.height);
    glUniform1f(myTimeLoc, CFAbsoluteTimeGetCurrent());
    glDrawArrays(GL_TRIANGLES, 0, 6);
    glEnable(GL_DEPTH_TEST); // Restore default
    glBindVertexArrayAPPLE(0); // Unbind
}
```

Mixing OpenGL and Scene Kit

Scene delegate rendering code example

```
- (void)renderer:(id <SCNSceneRenderer>)aRenderer
 willRenderScene:(SCNScene *)scene
          atTime:(NSTimeInterval)time
{
    // Custom OpenGL code
    glBindVertexArrayAPPLE(myVAO);
    glDisable(GL_DEPTH_TEST);
    glUseProgram(myProgram);
    CGSize size = self.myView.frame.size;
    glUniform2f(myResolutionLoc, size.width, size.height);
    glUniform1f(myTimeLoc, CFAbsoluteTimeGetCurrent());
    glDrawArrays(GL_TRIANGLES, 0, 6);
    glEnable(GL_DEPTH_TEST); // Restore default
    glBindVertexArrayAPPLE(0); // Unbind
}
```

Mixing OpenGL and Scene Kit

Scene delegate rendering code example

```
- (void)renderer:(id <SCNSceneRenderer>)aRenderer
 willRenderScene:(SCNScene *)scene
          atTime:(NSTimeInterval)time
{
    // Custom OpenGL code
    glBindVertexArrayAPPLE(myVAO);
    glDisable(GL_DEPTH_TEST);
    glUseProgram(myProgram);
    CGSize size = self.myView.frame.size;
    glUniform2f(myResolutionLoc, size.width, size.height);
    glUniform1f(myTimeLoc, CFAbsoluteTimeGetCurrent());
    glDrawArrays(GL_TRIANGLES, 0, 6);
    glEnable(GL_DEPTH_TEST); // Restore default
    glBindVertexArrayAPPLE(0); // Unbind
}
```

Extending Scene Kit with OpenGL

Node delegate rendering

- Custom OpenGL code per node
- Overrides Scene Kit's rendering
- Transform and geometry information are provided by Scene Kit

Extending Scene Kit with OpenGL

Node delegate rendering

Extending Scene Kit with OpenGL

Node delegate rendering code example

```
aNode.rendererDelegate = self;


- (void)renderNode:(SCNNode *)node
          renderer:(SCNRenderer *)renderer
         arguments:(NSDictionary *)arguments
{
   // Draw custom particle system using OpenGL
}
```

Extending Scene Kit with OpenGL

Material custom program

- Custom GLSL code per material
- Overrides Scene Kit's rendering
- Geometry attributes are provided by Scene Kit
- Transform uniforms also provided

Mixing OpenGL and Scene Kit

Material custom program

Extending Scene Kit with OpenGL

Material program creation code example

```
SCNProgram *myProgram = [[SCNProgram alloc] init];


NSString *vertexShader = @"…"
NSString *fragmentShader = @"…"


myProgram.vertexShader   = vertexShader;
myProgram.fragmentShader = fragmentShader;


aMaterial.program = myProgram;
```

Extending Scene Kit with OpenGL

Material program creation code example

```
SCNProgram *myProgram = [[SCNProgram alloc] init];

NSString *vertexShader = @"…"
NSString *fragmentShader = @"…"

myProgram.vertexShader   = vertexShader;
myProgram.fragmentShader = fragmentShader;

aMaterial.program = myProgram;
```

Extending Scene Kit with OpenGL

Material program creation code example

```
SCNProgram *myProgram = [[SCNProgram alloc] init];


NSString *vertexShader = @"…"
NSString *fragmentShader = @"…"


myProgram.vertexShader   = vertexShader;
myProgram.fragmentShader = fragmentShader;


aMaterial.program = myProgram;
```

Extending Scene Kit with OpenGL

Material program creation code example

```
SCNProgram *myProgram = [[SCNProgram alloc] init];


NSString *vertexShader = @"…"
NSString *fragmentShader = @"…"


myProgram.vertexShader   = vertexShader;
myProgram.fragmentShader = fragmentShader;


aMaterial.program = myProgram;
```

Extending Scene Kit with OpenGL

Material program shaders example

```
attribute vec3 a_position;
attribute vec3 a_normal;
uniform mat4 u_mvpMatrix;
uniform mat4 u_normalMatrix;

varying vec3 v_normal;
varying vec2 v_uv;

void main() {
    vec4 normals   = u_normalMatrix * vec4(a_normal.xyz,
                     1.0);
    v_normal       = normalize(normals.xyz);
    v_uv           = normals.xy;
    gl_Position    = u_mvpMatrix * v_position;
}
```

Vertex Shader

```
varying vec3 v_normal; //in view space, transformed by the VS
varying vec2 v_uv;
uniform vec2 u_fresnel; // x:amount, y:power

float fresnel(vec3 n, vec3 v) {
    float dp = clamp(dot(v, n), 0.0, 1.0);
    return u_fresnel.x + (1.0 - u_fresnel.x) * pow(1.0 - dp,
           u_fresnel.y);
}

void main() {
    float f = fresnel(v_normal, vec3(0.0, 0.0, 1.0));
    gl_FragColor = vec4(v_uv.x * f, v_uv.y * f, f, 1.0);
}
```

Fragment Shader

Extending Scene Kit with OpenGL

Material program shaders example

```
attribute vec3 a_position;
attribute vec3 a_normal;
uniform mat4 u_mvpMatrix;
uniform mat4 u_normalMatrix;

varying vec3 v_normal;
varying vec2 v_uv;

void main() {
    vec4 normals   = u_normalMatrix * vec4(a_normal.xyz,
                     1.0);
    v_normal       = normalize(normals.xyz);
    v_uv           = normals.xy;
    gl_Position    = u_mvpMatrix * v_position;
}
```

Vertex Shader

```
varying vec3 v_normal; //in view space, transformed by the VS
varying vec2 v_uv;
uniform vec2 u_fresnel; // x:amount, y:power

float fresnel(vec3 n, vec3 v) {
    float dp = clamp(dot(v, n), 0.0, 1.0);
    return u_fresnel.x + (1.0 - u_fresnel.x) * pow(1.0 - dp,
           u_fresnel.y);
}

void main() {
    float f = fresnel(v_normal, vec3(0.0, 0.0, 1.0));
    gl_FragColor = vec4(v_uv.x * f, v_uv.y * f, f, 1.0);
}
```

Fragment Shader

Extending Scene Kit with OpenGL

Binding program semantics

```
attribute vec3 a_position;
attribute vec3 a_normal;
uniform mat4 u_mvpMatrix;
uniform mat4 u_normalMatrix;
```

```
uniform vec2 u_fresnel; // x:amount, y:power
```

Extending Scene Kit with OpenGL

Binding program semantics

```
attribute vec3 a_position;
attribute vec3 a_normal;
uniform mat4 u_mvpMatrix;
uniform mat4 u_normalMatrix;
```

```
uniform vec2 u_fresnel; // x:amount, y:power
```

Extending Scene Kit with OpenGL

Binding program semantics

```
attribute vec3 a_position;
attribute vec3 a_normal;
uniform mat4 u_mvpMatrix;
uniform mat4 u_normalMatrix;
```

```
uniform vec2 u_fresnel; // x:amount, y:power
```

```
[myProgram setSemantic:SCNGeometrySourceSemanticVertex forSymbol:@"a_position" options:nil];
[myProgram setSemantic:SCNGeometrySourceSemanticNormal forSymbol:@"a_normal" options:nil];
[myProgram setSemantic:SCNModelViewProjectionTransform forSymbol:@"u_mvpMatrix" options:nil];
[myProgram setSemantic:SCNNormalTransform forSymbol:@"u_normalMatrix" options:nil];
```

Extending Scene Kit with OpenGL

Binding program semantics

```
attribute vec3 a_position;
attribute vec3 a_normal;
uniform mat4 u_mvpMatrix;
uniform mat4 u_normalMatrix;
```

```
uniform vec2 u_fresnel; // x:amount, y:power
```

```
- (BOOL)   program:(SCNProgram *)program
bindValueForSymbol:(NSString *)symbol
        atLocation:(unsigned int)location
         programID:(unsigned int)programID
          renderer:(SCNRenderer *)renderer
{
    // no need to bind the program (already done)
    if ([symbol isEqualToString:@"u_Fresnel"])
        glUniform2f(location, 0.1, 1.5); // amount, exponent
}
```

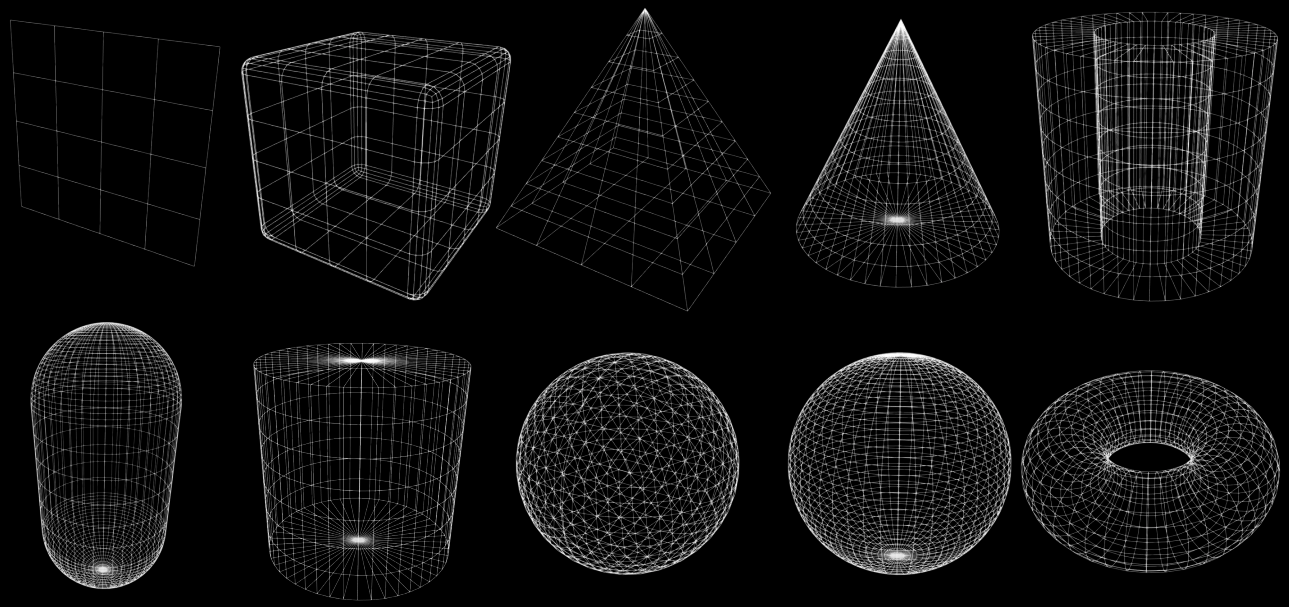
Creating Geometries

Creating Geometries

Built-in parametric primitives

Creating Geometries

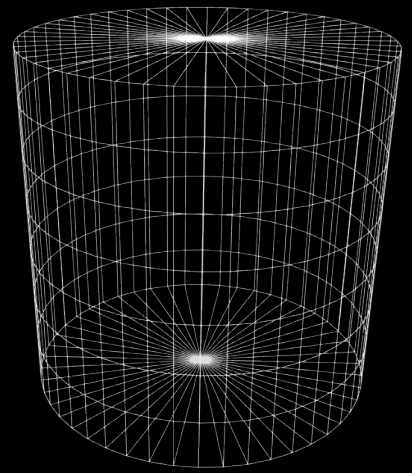
Built-in parametric primitives



Creating Geometries

Built-in parametric primitives code example

```
// Create a sample cylinder primitive
SCNCylinder *myCylinder =
[SCNCylinder cylinderWithRadius:1.0
                         height:2.0];
myCylinder.radialSegmentCount = 32;
myCylinder.heightSegmentCount = 4;
myNode.geometry = myCylinder;
```



Creating Geometries

Built-in 3D text

```
// Create a SCNText
SCNText *aText = [SCNText text];
aText.font = [NSFont fontWithName:@"Avenir Next Heavy" size:40];
aText.string = @"WWDC";
aText.extrusionDepth = 10.0;
aText.materials = someMaterials;
aTextNode.geometry = aText;
```

Creating Geometries

Built-in 3D text

```
// Create a SCNText
SCNText *aText = [SCNText text];
aText.font = [NSFont fontWithName:@"Avenir Next Heavy" size:40];
aText.string = @"WWDC";
aText.extrusionDepth = 10.0;
aText.materials = someMaterials;
aTextNode.geometry = aText;
```

Creating Geometries

Built-in reflective floor

```
// Create a SCNFloor
SCNFloor *myFloor = [SCNFloor floor];
myFloor.reflectivity = 0.8;
myFloor.reflectionFalloffStart = 0.0;
myFloor.reflectionFalloffEnd = 1.0;
myFloor.firstMaterial.diffuse.contents = [NSColor blackColor];
myFloorNode.geometry = myFloor;
```

Creating Geometries

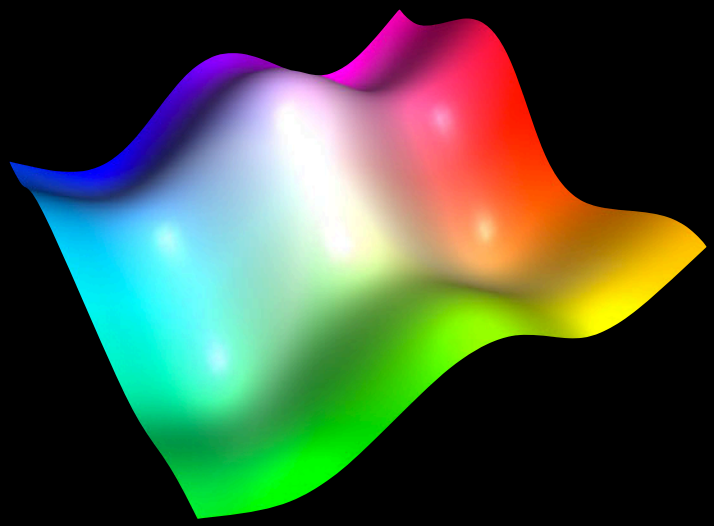
Custom geometries

- Custom vertices, normals, and texture coordinates

Creating Geometries

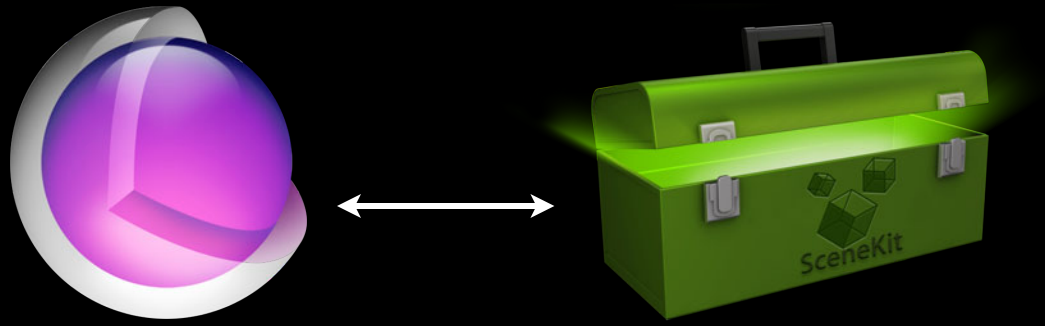
Custom geometries

- Custom vertices, normals, and texture coordinates



Mixing Core Animation and Scene Kit

- Scene Kit inside a Core Animation layer
- A Core Animation layer inside Scene Kit



Mixing Core Animation and Scene Kit

Scene Kit inside a Core Animation layer

```
// Create a SCNLayer and set a scene to it.
SCNLayer *mySCNLayer = [SCNLayer layer];
mySCNLayer.scene = aScene;

// Insert the SCNLayer into a layer tree.
[aLayer addSublayer:mySCNLayer];
```

Mixing Core Animation and Scene Kit

A Core Animation layer inside Scene Kit

```
// Map a layer tree on a 3D object.
myNode.geometry.firstMaterial.diffuse.contents = aLayerTree;
```

Demo

Going Beyond

Recap

- Animations
- Extending Scene Kit with OpenGL
- Creating Geometries
- Mixing Core Animation with Scene Kit

More Information

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

Mike Jurewitz
Developer Tools and Performance Evangelist
jurewitz@apple.com

Documentation
Scene Kit API reference documentation
http://developer.apple.com/ue

Apple Developer Forums
http://devforums.apple.com

Labs

Scene Kit Lab	Location Tuesday 3:00PM


WWDC2012