iPhone Application Programming
Lecture 7: Rendering, Metal, Sprite kit, Scene kit

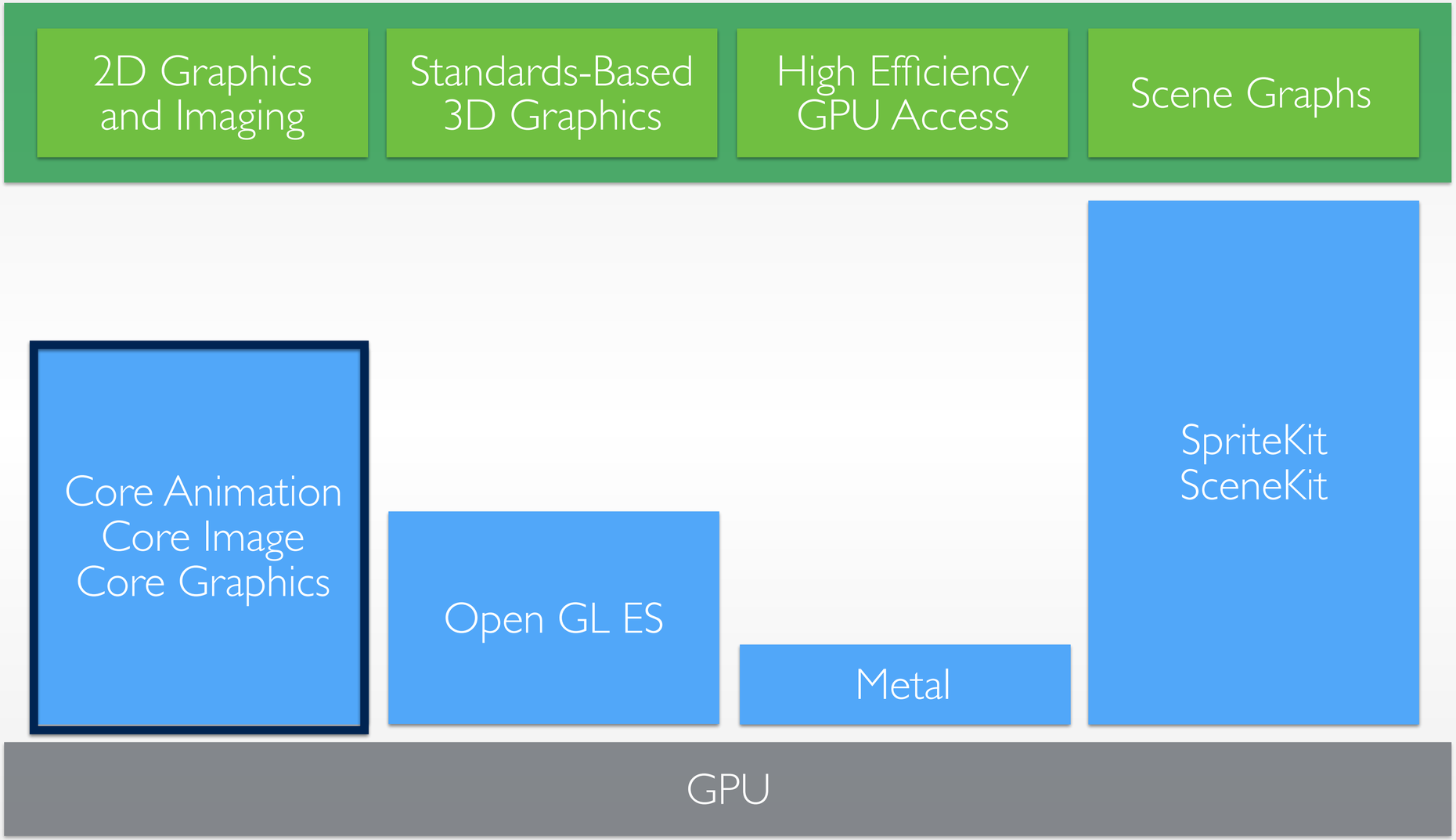
Simon Voelker
Media Computing Group
RWTH Aachen University

Winter Semester 2015/2016

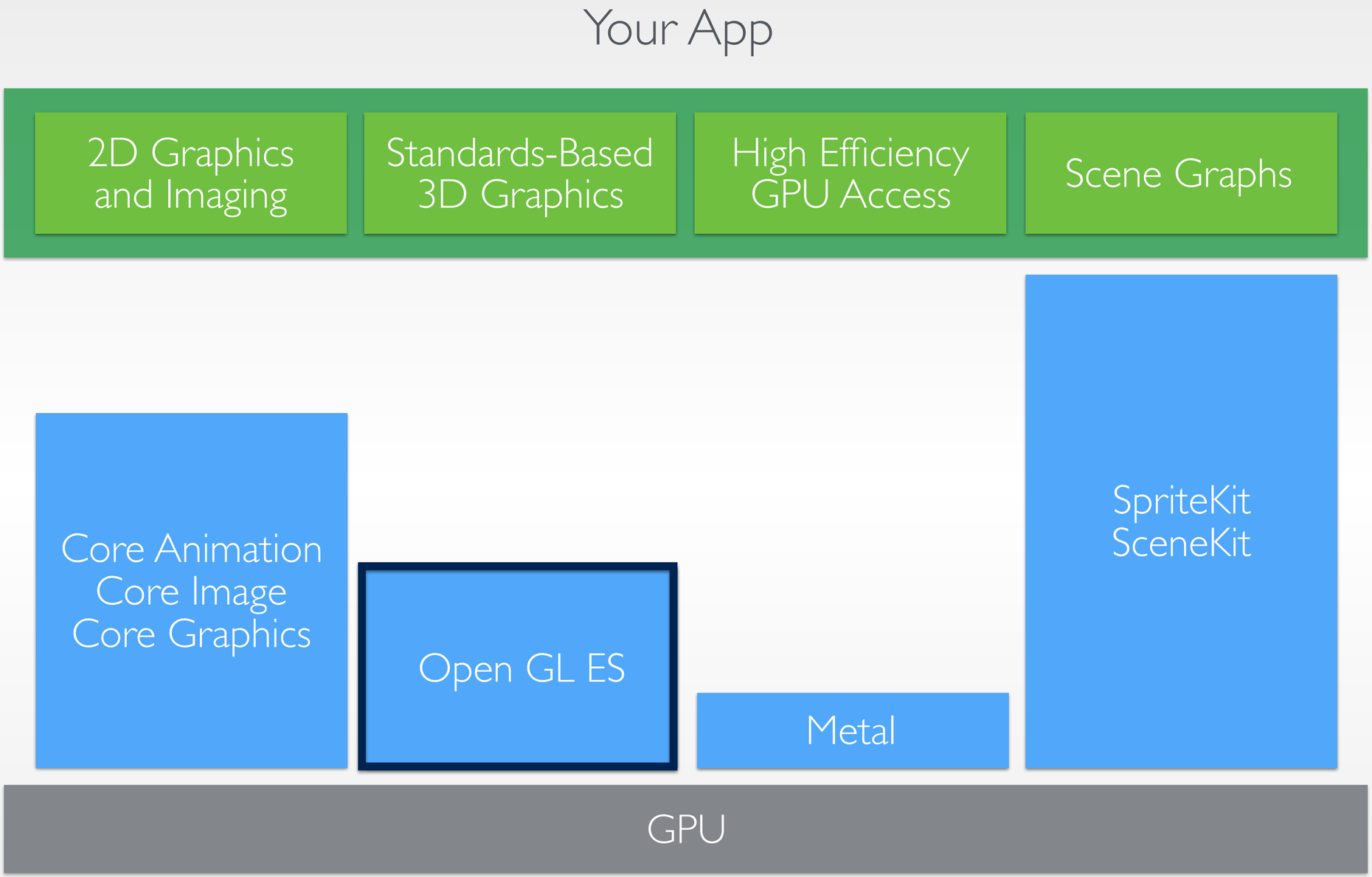
http://hci.rwth-aachen.de/iphone

Media Computing Group | RWTH AACHEN UNIVERSITY

Your App

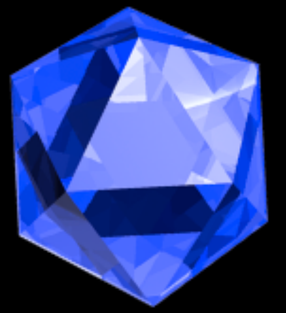


Apple, WWDC 2014



Apple, WWDC 2014


T-Mobile
17:19



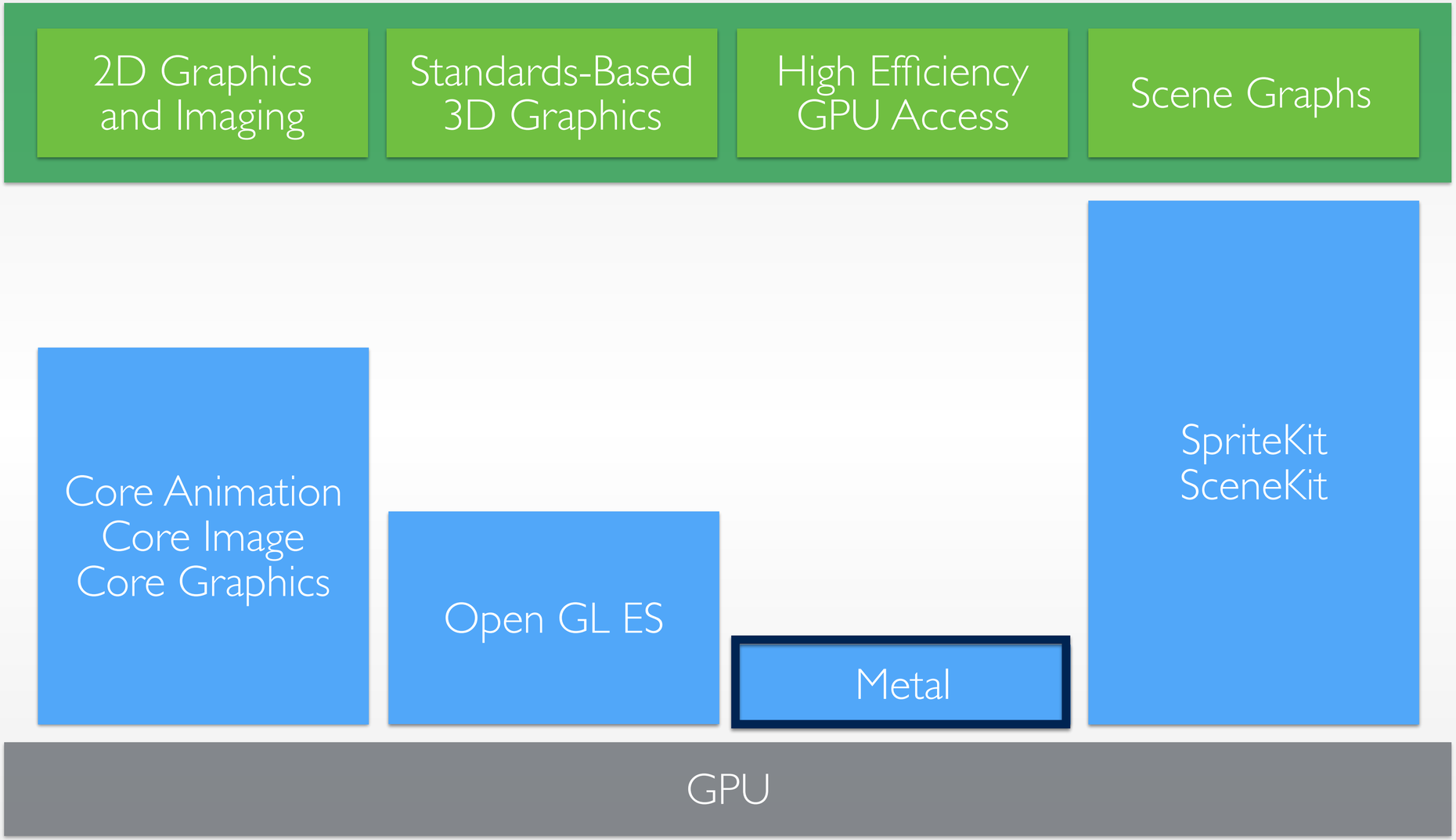
OpenGL ES

- 3D graphics rendering engine
- OpenGL with reduced instruction set
- Cross-platform
- C based
- OpenGL is pixel based
- Version 1.1: Fixed rendering pipeline
- Version 2.0: Shader-based rendering pipeline

GLKit

- Introduced iOS 5
- More comfort for basic rendering tasks
- Helper classes
  - GLKView & GLKViewController
  - GLKTextureLoader
  - GLKEffect
  - Math Library
  - Lighting
- For More details: iPhone Lecture 6 (2012/2013)

Your App



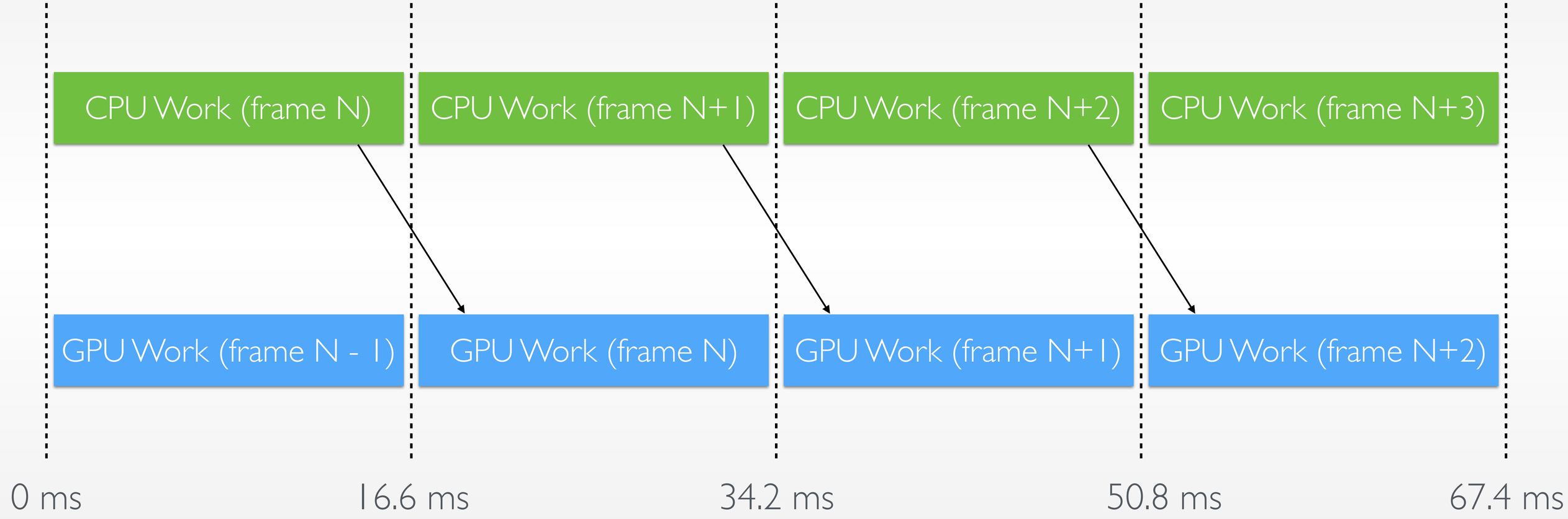
Apple, WWDC 2014

Metal

- Highly optimised rendering pipeline (OpenGL overhead reduced)
- C++
- Precompiled shaders
- Multithreading
- Speed increase by reducing CPU load
- Works only on A7 and A8 processors (iPhone 5s)

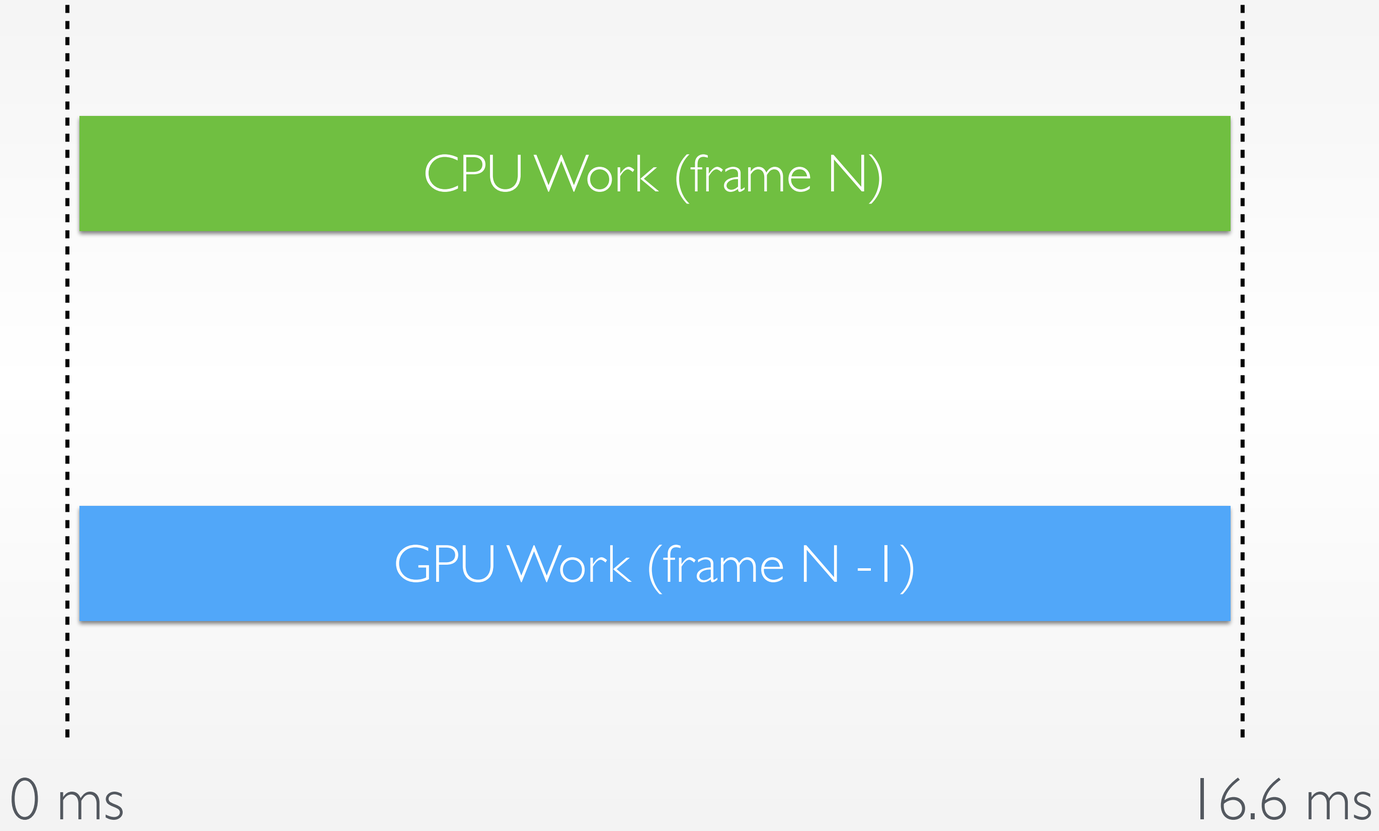
Improve Rendering Performance

60 FPS (16.6 ms/frame)



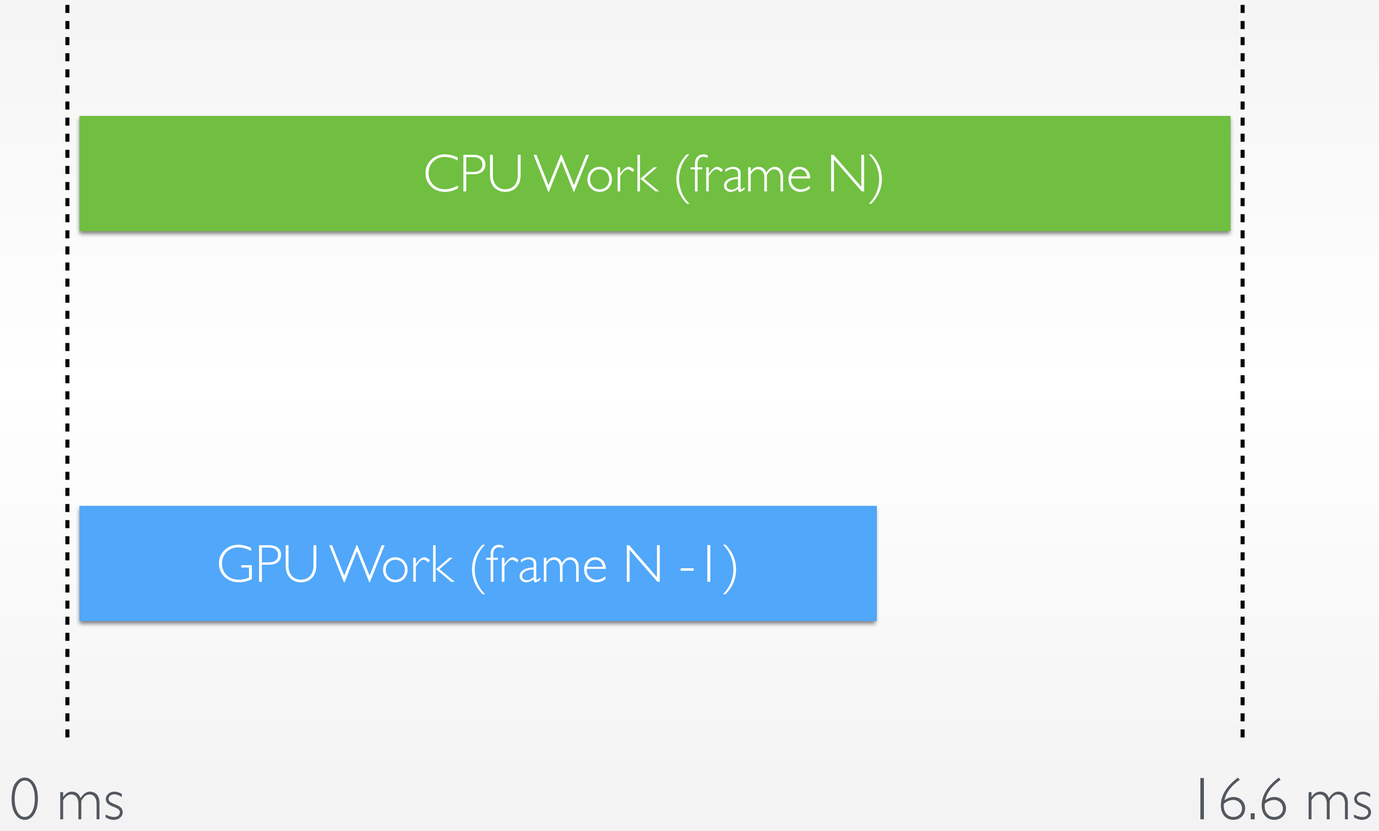
Improve Rendering Performance

60 FPS (16.6 ms/frame)



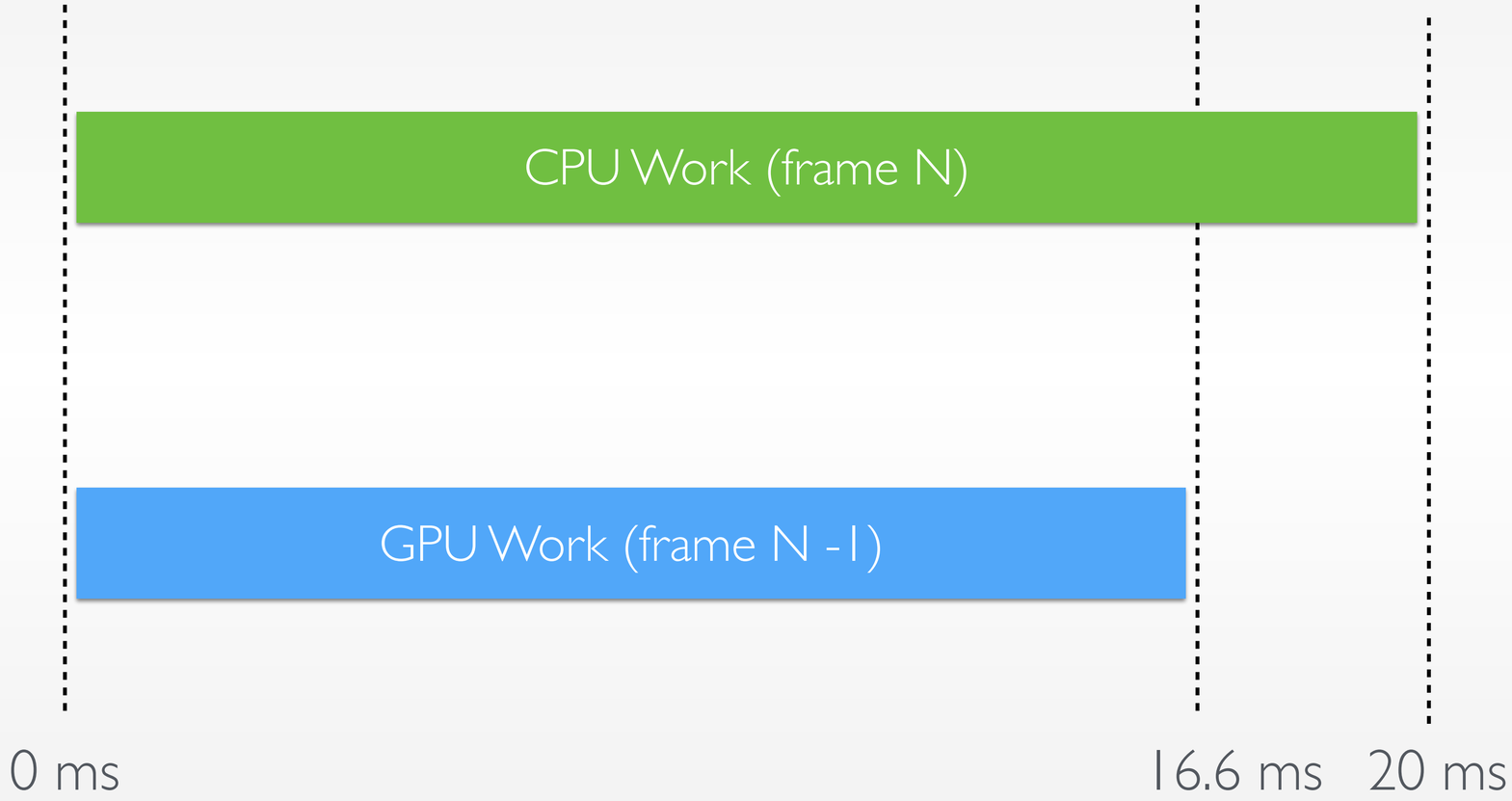
Improve Rendering Performance

60 FPS (16.6 ms/frame)



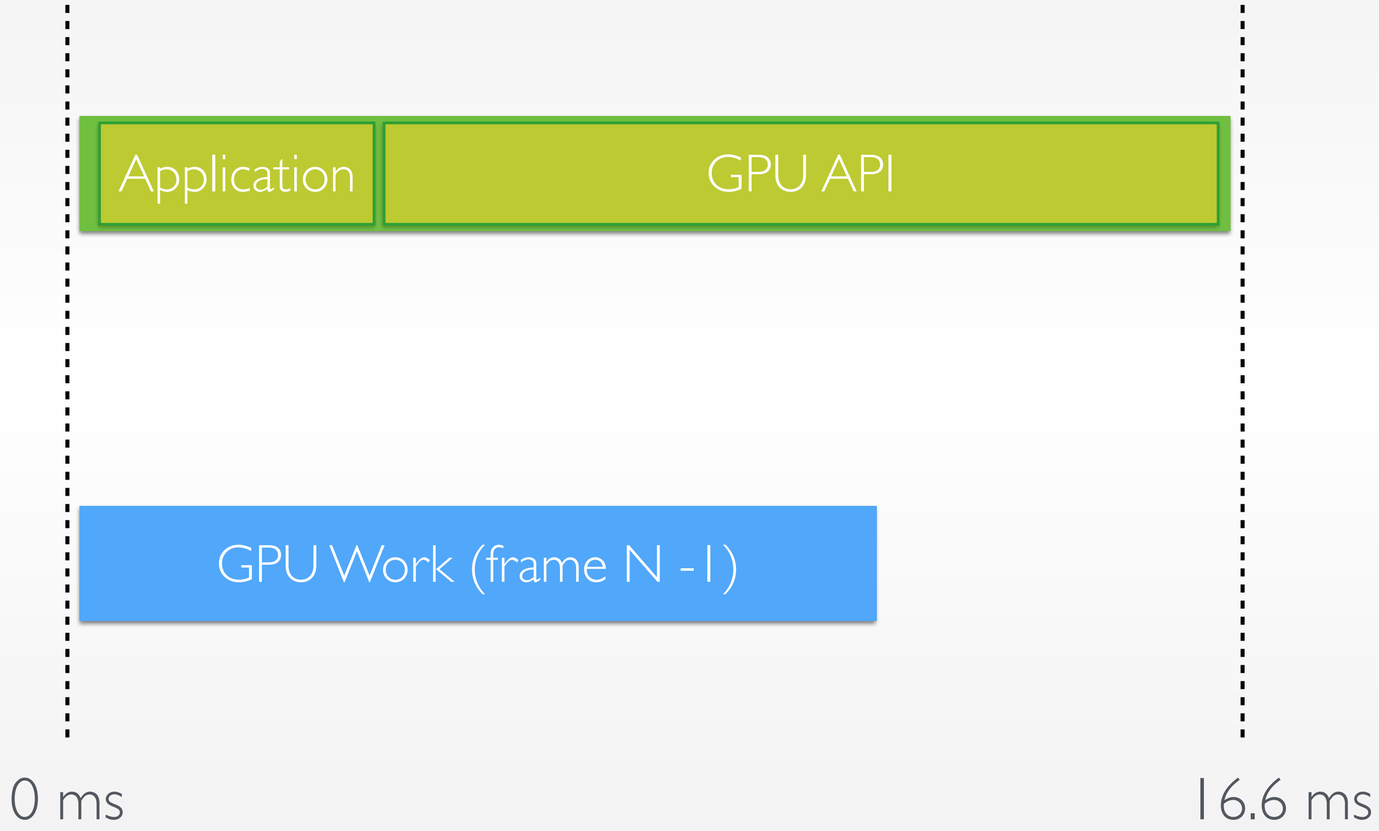
Improve Rendering Performance

60 FPS (16.6 ms/frame)



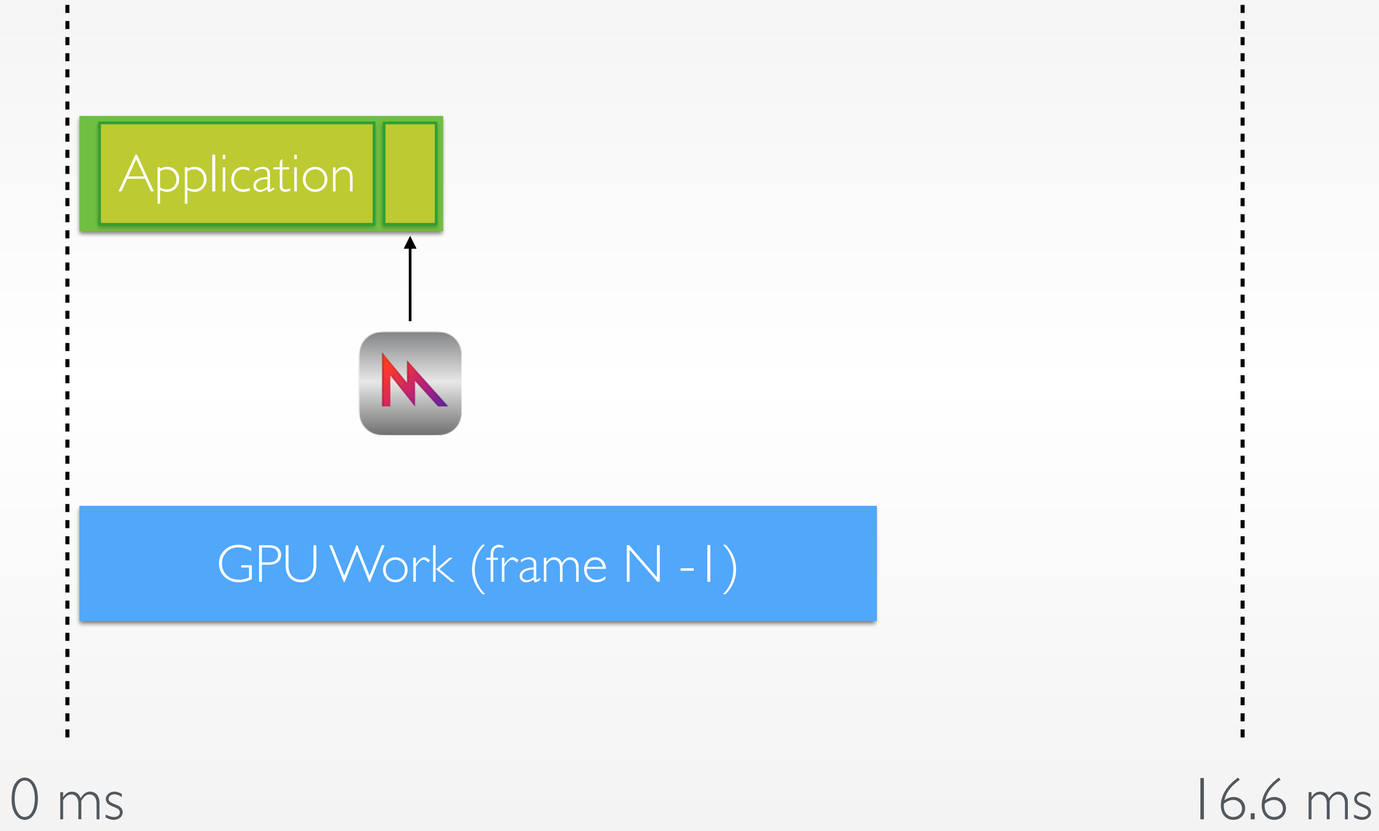
Improve Rendering Performance

60 FPS (16.6 ms/frame)



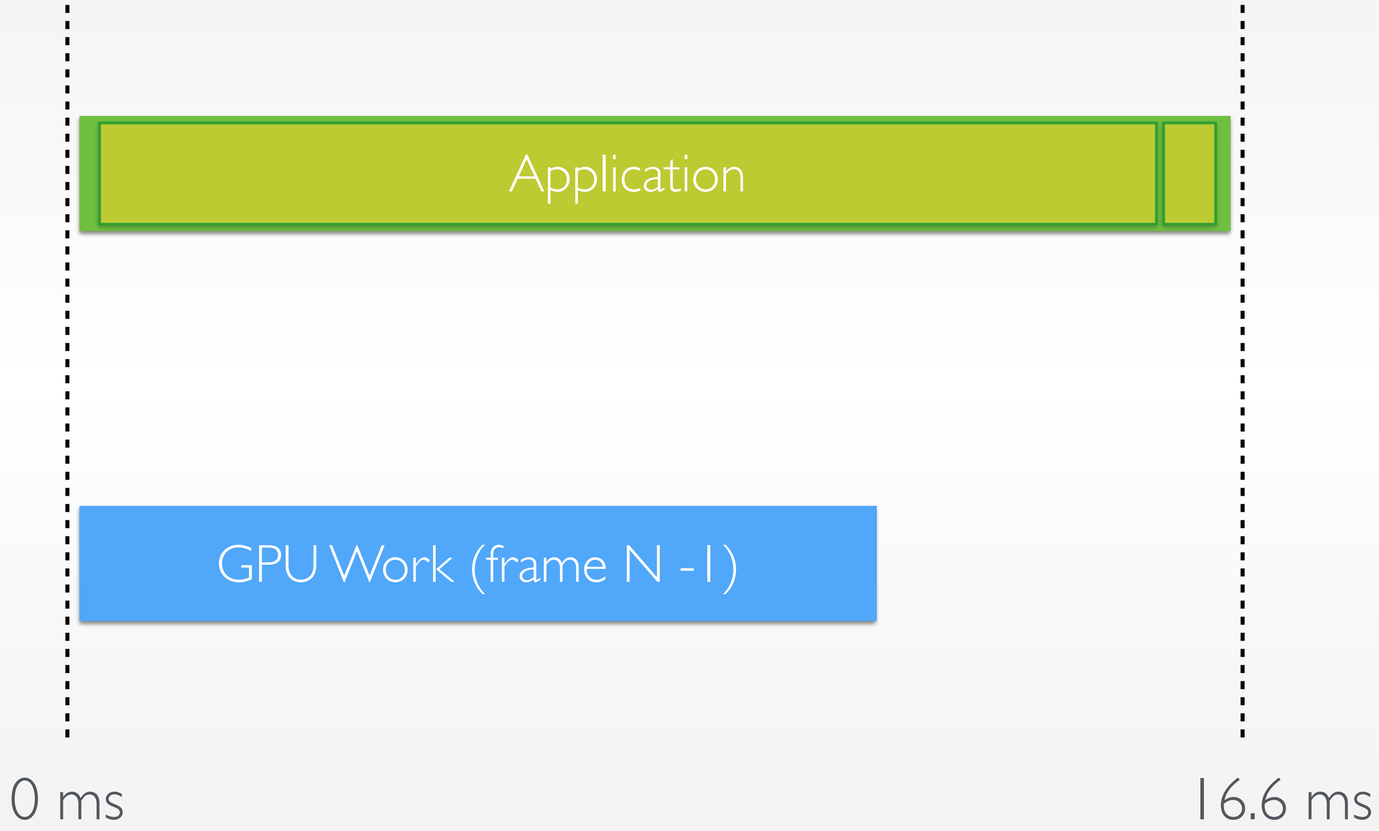
Improve Rendering Performance

60 FPS (16.6 ms/frame)



Improve Rendering Performance

60 FPS (16.6 ms/frame)



Expansive GPU API

- State validation
  - Check if API use is valid
  - Encode API state to hardware state
- Shader compilation
  - Run-time generation of shader code
- Transferring work to GPU
  - Resources
  - Commands

Expansive GPU API

When	Frequency	Before Metal	Metal
Application build	never		
Content loading	Rare		
Draw time	every frame		

Apple, WWDC 2013

Expansive GPU API

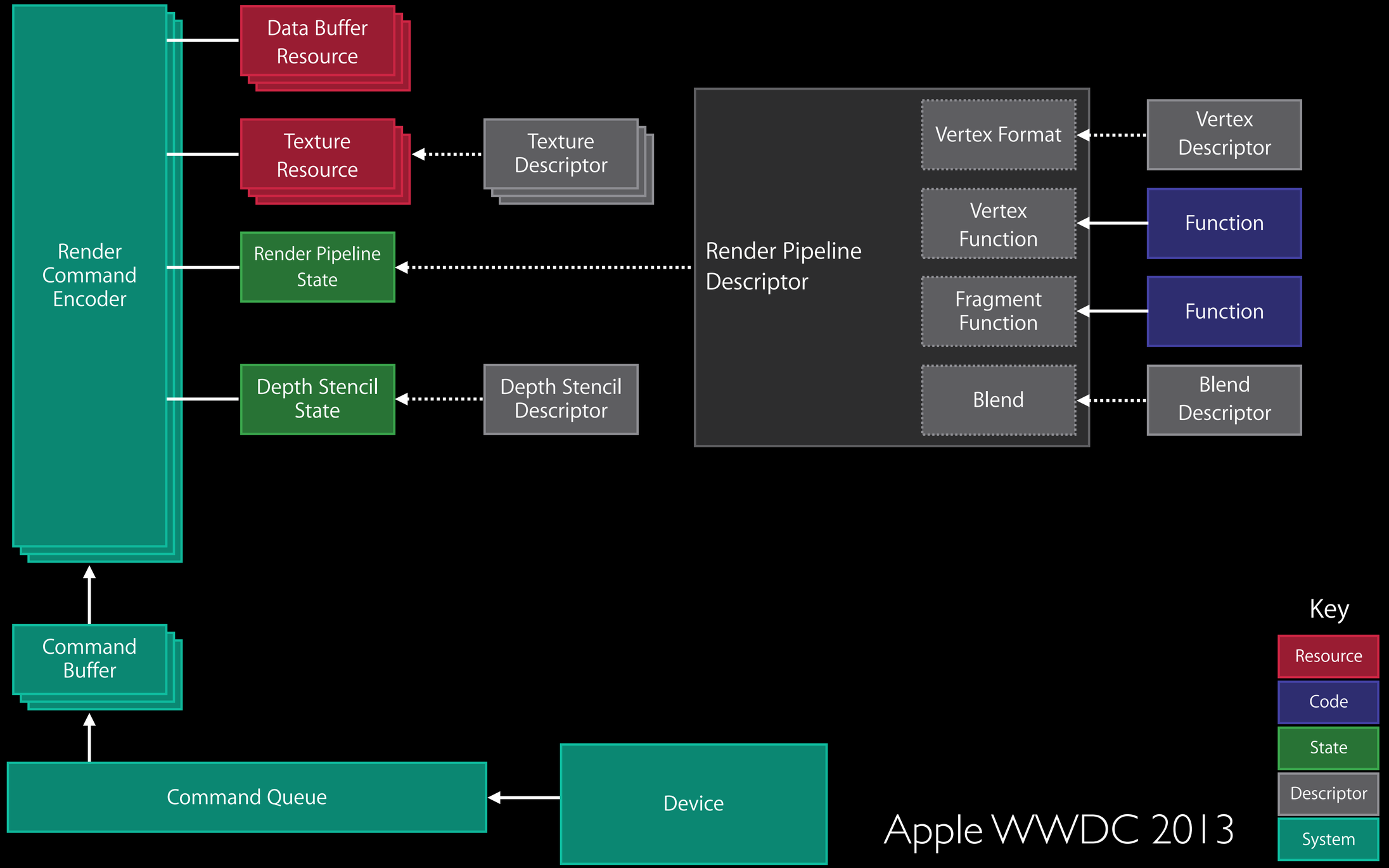
When	Frequency	Before Metal	Metal
Application build	never		
Content loading	Rare		
Draw time	every frame	State validation Shader compilation Start work on the GPU	

Apple, WWDC 2013

Expansive GPU API

When	Frequency	Before Metal	Metal
Application build	never		Shader compilation
Content loading	Rare		State validation
Draw time	every frame	State validation Shader compilation Start work on the GPU	Start work on the GPU

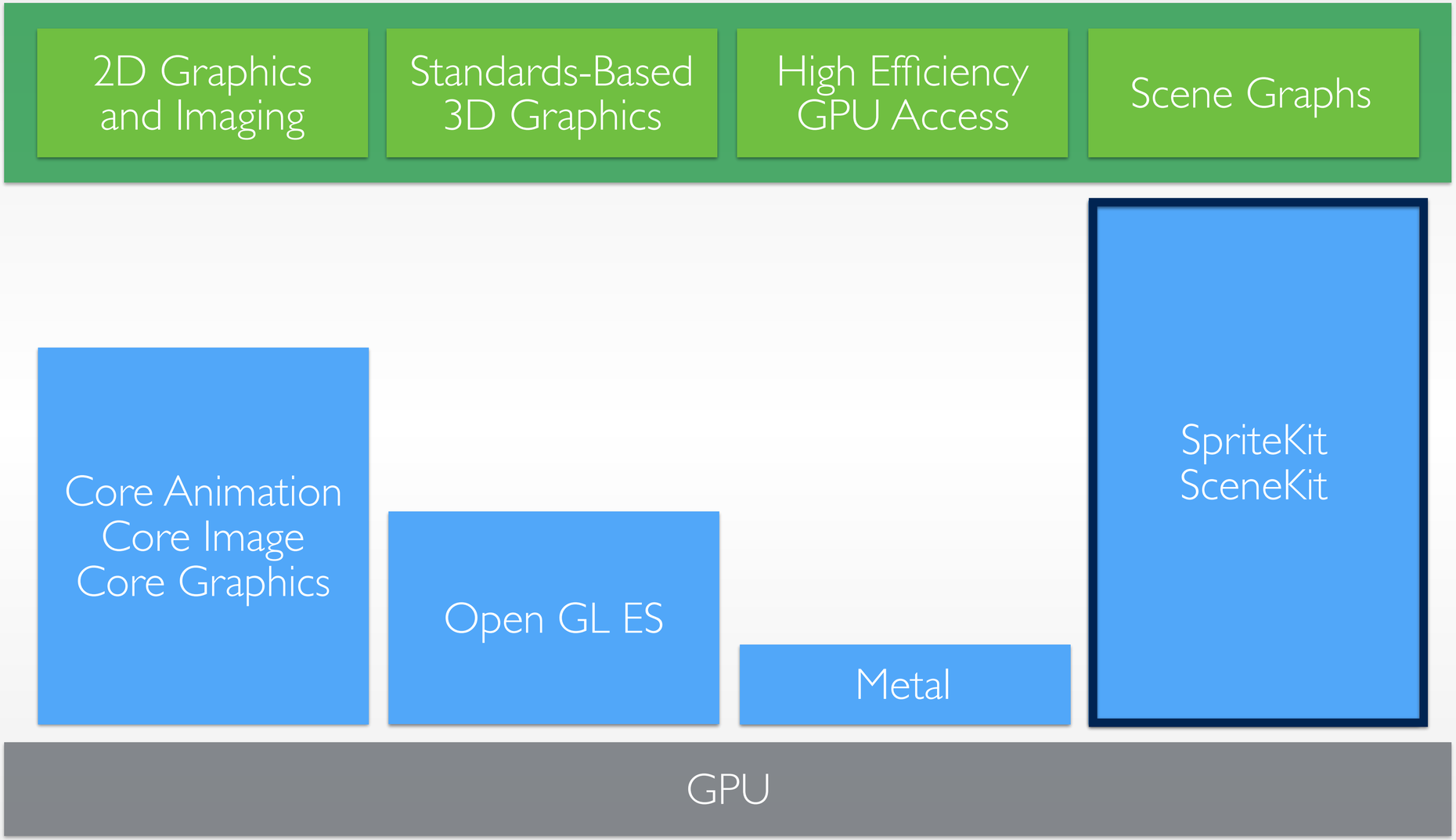
Apple, WWDC 2013



Apple WWDC 2013

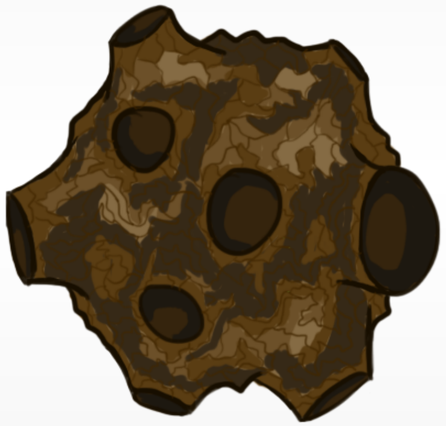
Metal Demo

Your App



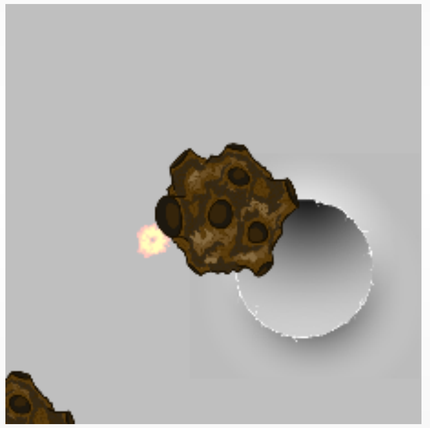
Apple, WWDC 2014

Sprite Kit

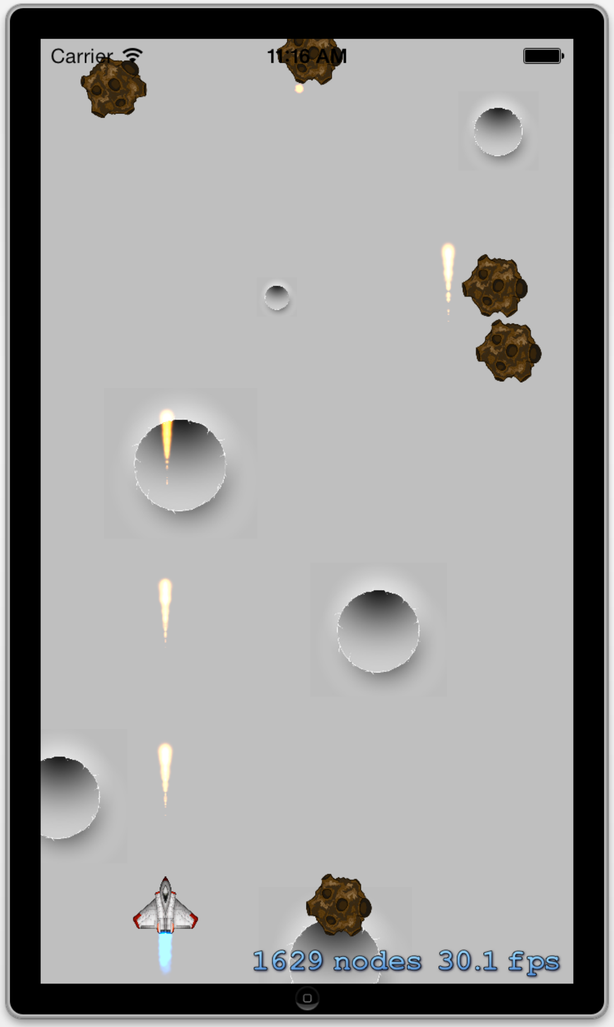


Objects

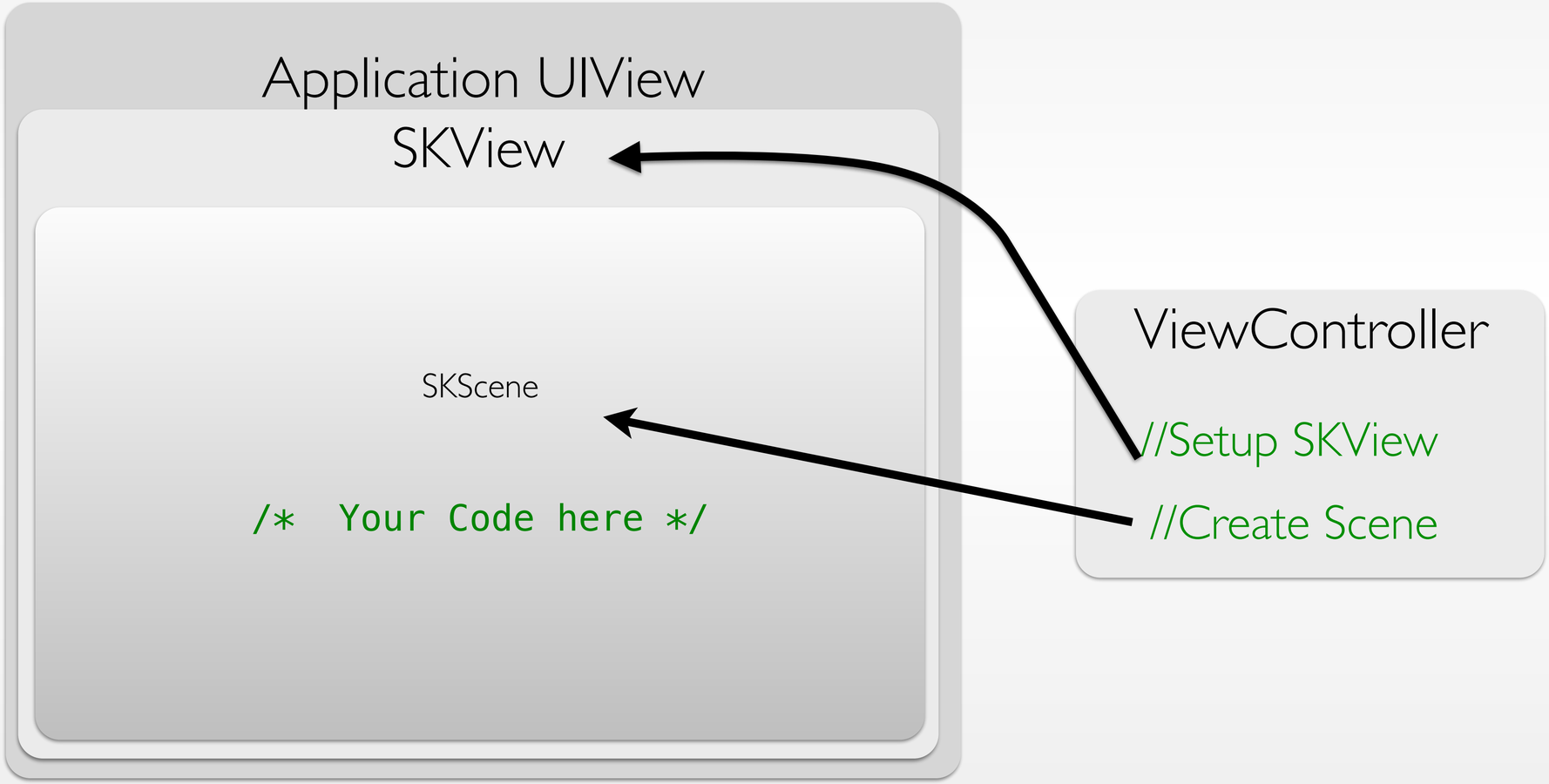
Actions



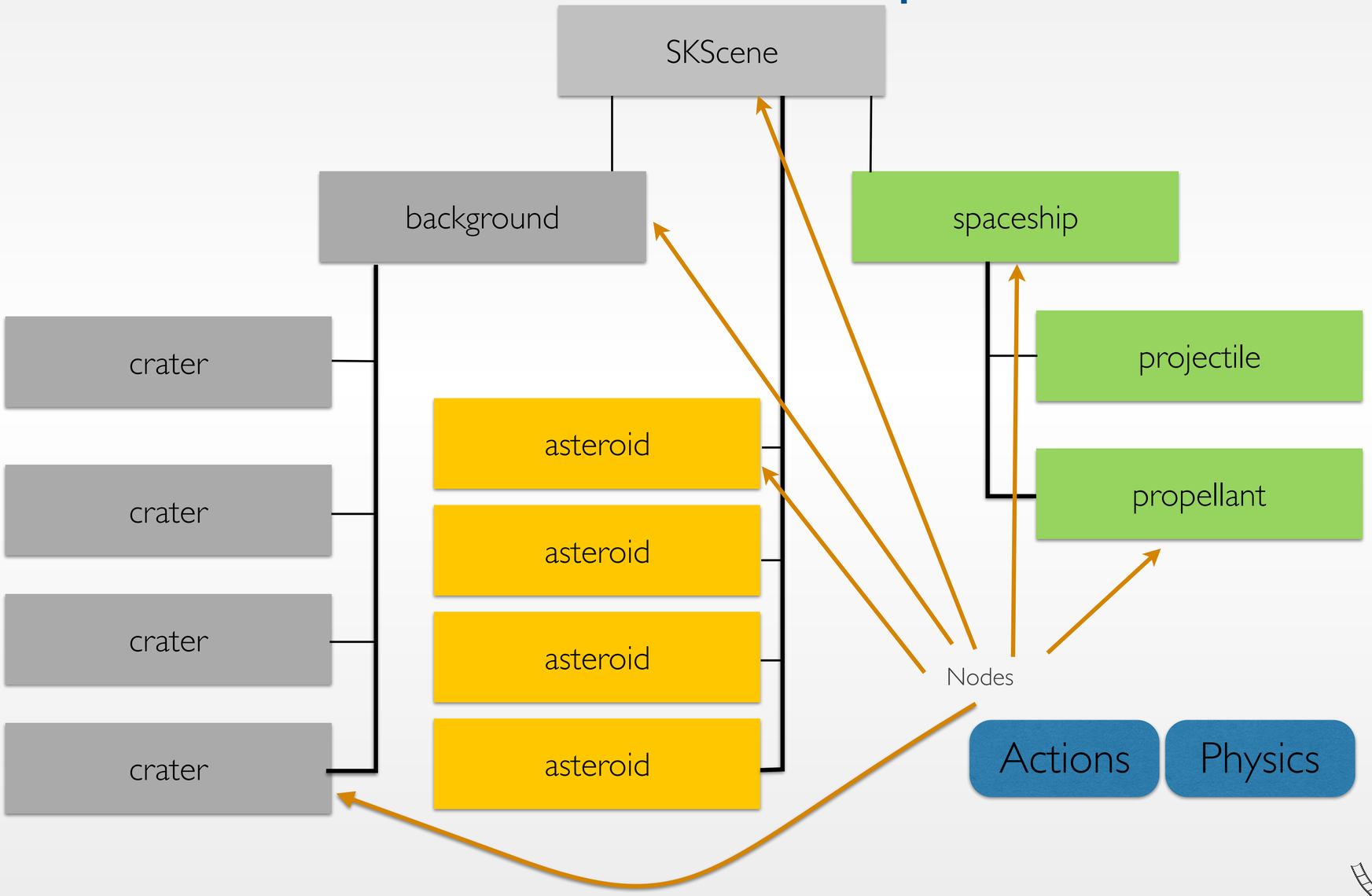
Physics



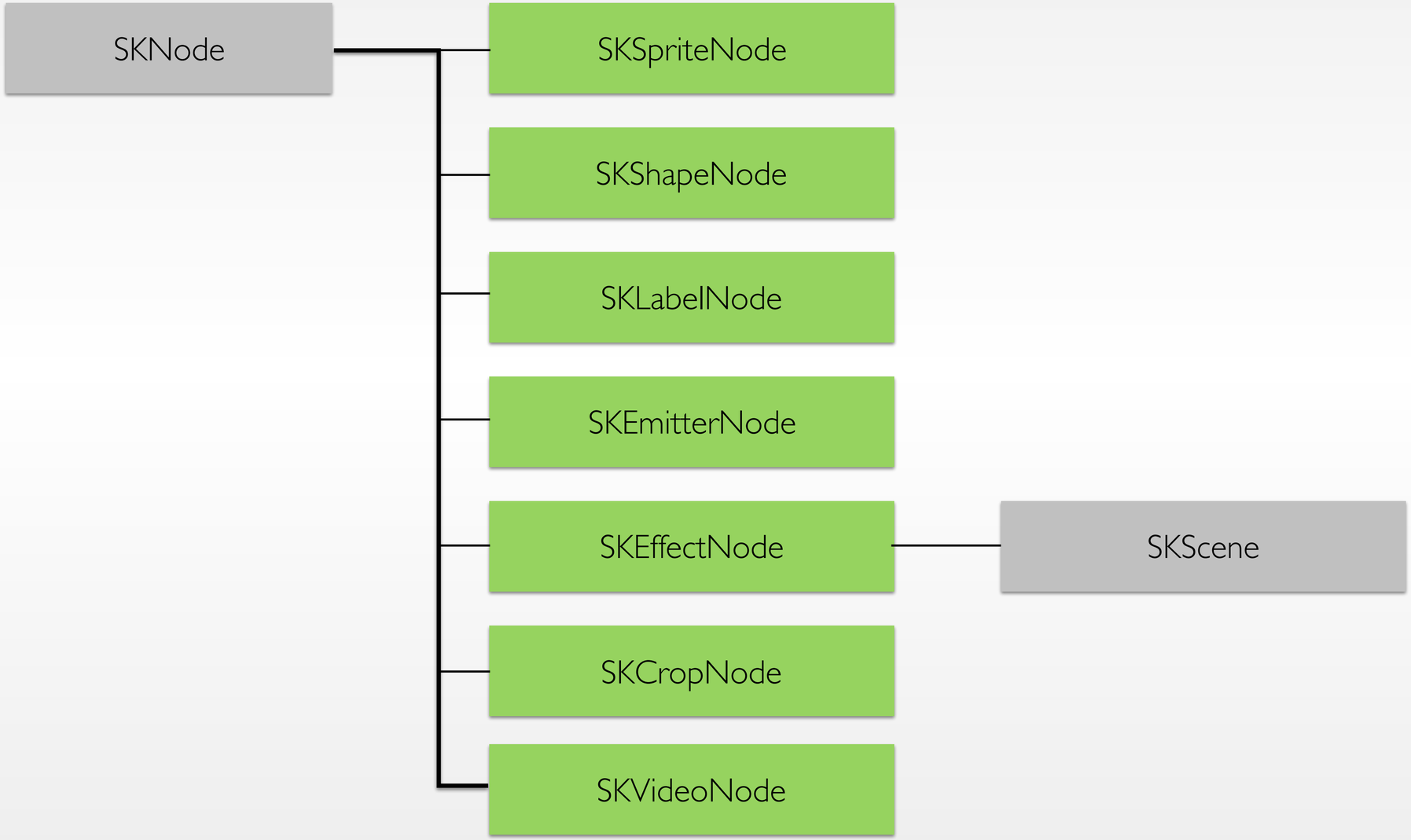
Root Object: SKScene



Scene Graph

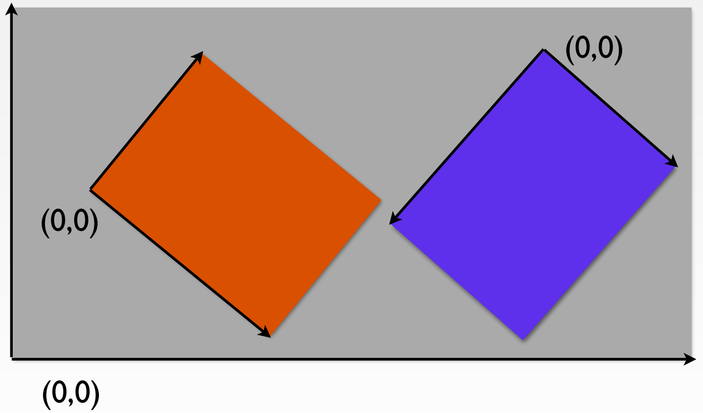


Sprite Kit Nodes



SKNode

- Basic node (used for grouping)
- Position, rotation, scale
- zPosition




```
//Hit Test
node.containsPoint(aCGPoint)

//Converts a point from the coordinate system
node.convertPoint(aCGPoint, fromNode: aSKNode)


//Converts a point in this node's coordinate system
node.convertPoint(aCGPoint, toNode: aSKNode)
```

SKSpriteNode

solid color

texture

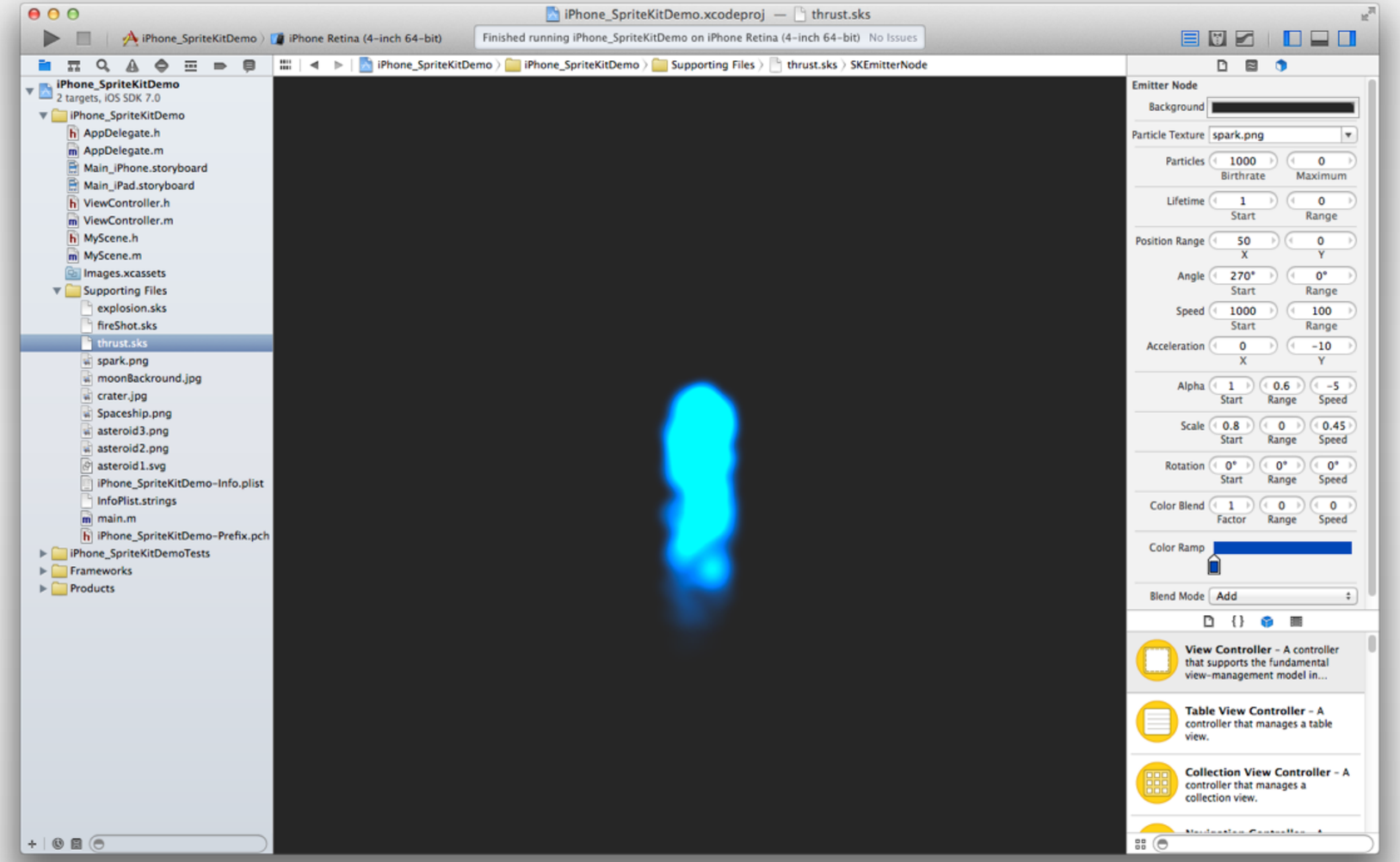
color blending

```
let sprite = SKSpriteNode(color: SKColor.greenColor(), size: CGSizeMake(100, 100))


let asteroid = SKSpriteNode(imageNamed:"asteroid.png")


asteroid.color = SKColor.greenColor()
asteroid.colorBlendFactor = 0.5
```

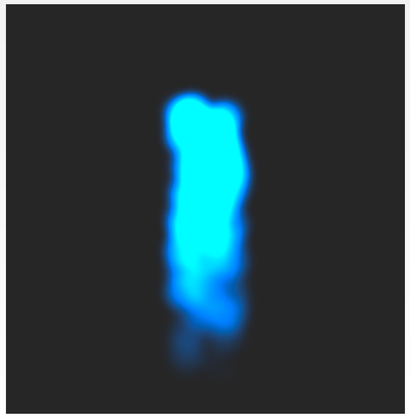
SKShapeNode

- Draws CGPath
- Stroke, Fill Color

Particle Editor



SKEmitterNode

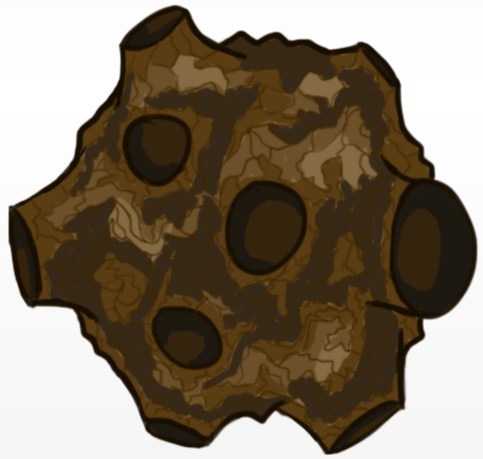


```
let thrust = SKEmitterNode(fileNamed: "thrust.sks")

thrust.position = CGPointMake(0, self.spaceship.size.height -10);

self.addChild(thrust!)

thrust.particleScale = 2;
thrust.particleScaleSpeed = -10;
```

SKCropNode

- Creates a mask the children
- Mask is defined as a SKNode

Mask

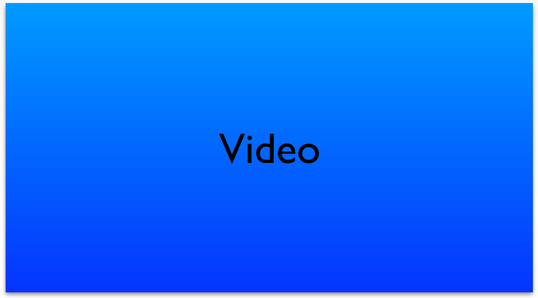


child

result

SKVideoNode

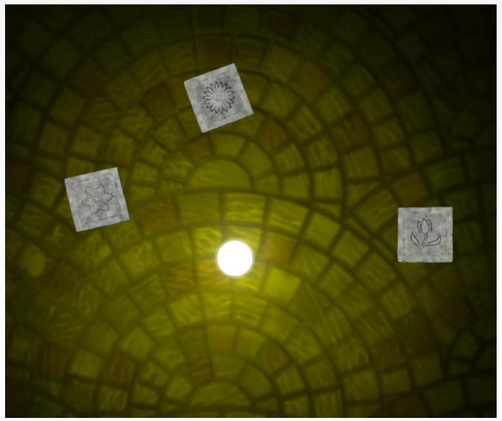
- Video as Node
- AVPPlayer (AVFoundation.framework)
- All the functionality from AVFoundation

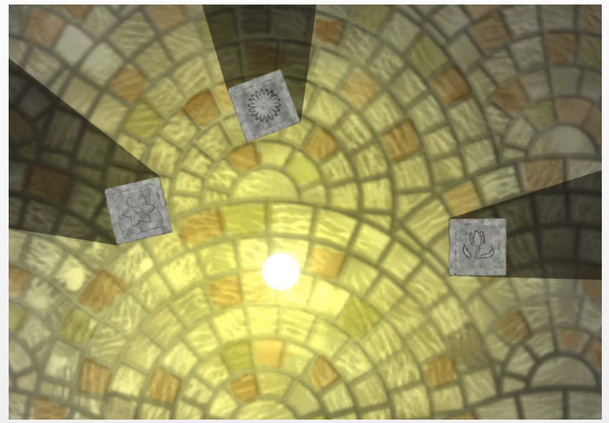



```
 let videoNode = SKVideoNode(fileNamed: “video”)

import AVFoundation
let videoNode = SKVideoNode(AVPlayer: player)

videoNode.play()
```

SKLightNode

- Creates Light
  - lightColor
  - shadowColor
  - ambientColor
- Can be set for each Node using a bitMask





SKEffectNode

- Applies CIFilter to its children
- CIFilter is a powerful Core Image filter
- Can be used on the entire Scene


```
 let filter = CIFilter(name: “CIColorInvert")

filter?.setDefaults()
```

SKEffectNode: CIFilter

- More then 100 different Filter
- Glow effects:
  - CIBloom

CIBloom

```
 let filter = CIFilter(name: “CIBloom")

self.filter = filter
self.shouldEnableEffects = true
```

SKShaders

- Basic OpenGL ES Fragment Shader
- Can be applied to
  - SKSpriteNodes, SKShapeNodes (stroke, fill), SKScenes..
- Two types of input variables:
  - Uniforms (value is the same across all shader instances)
  - Varying (value is different for each shader instances. e. g. pixel position)

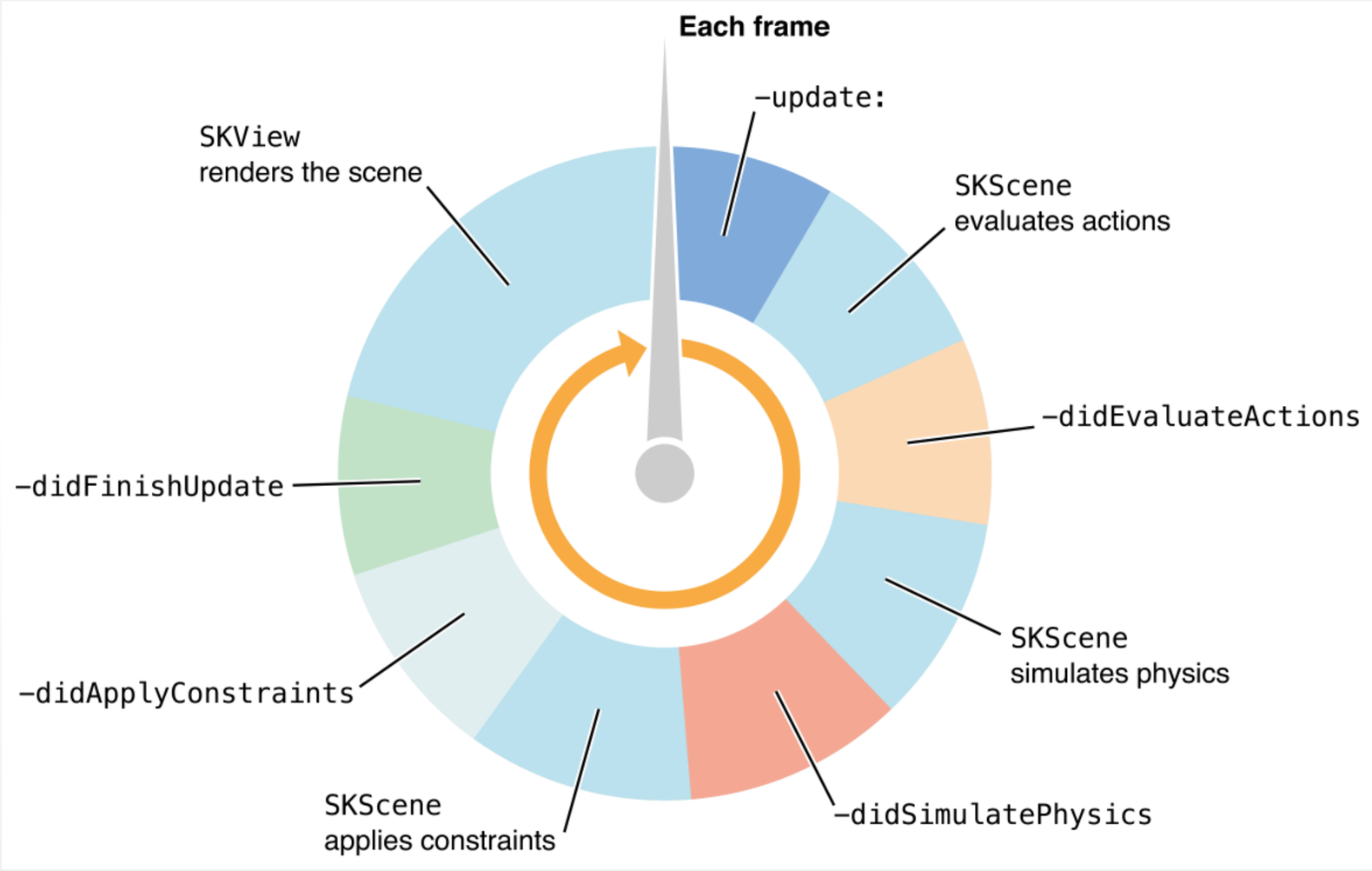
```
void main()
{
    gl_FragColor = SKDefaultShading();
}
```

```
void main()
{
    gl_FragColor = vec4(255,0,0,255);
}
```

Large library of Shaders: www.shadertoy.com

Sprite Kit Shader Demo

Sprite Kit Render Loop



Apple iOS 9 API

Sprite Kit: Actions

Simple Actions

Create Action

```
SKAction.moveTo(CGPoint(100,100), duration: 1.0)

SKAction.rotateByAngle(M_PI, duration: 1.0)
```

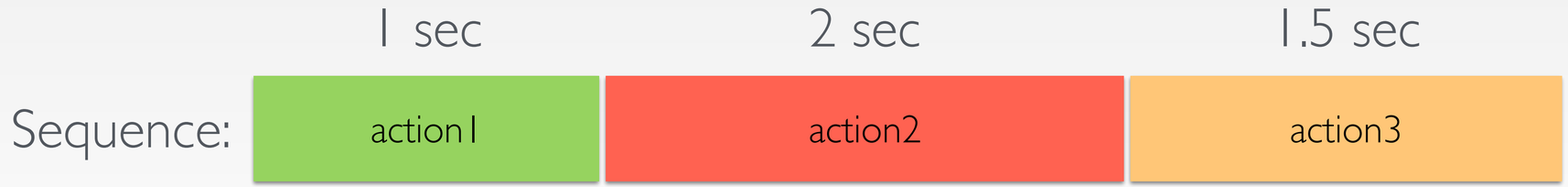
Move the spaceship

```
let action = SKAction.moveTo(CGPointMake(100,100), duration: 1.0)

spaceship.runAction(action)

spaceship.runAction(SKAction.moveTo(CGPointMake(100,100), duration: 1.0))
```

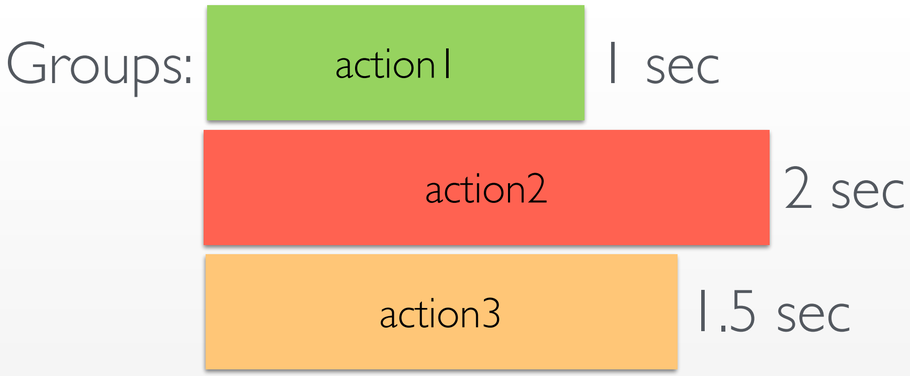
Repeating Actions

```
let action = SKAction.moveTo(CGPointMake(100,100), duration: 1.0)

shape.runAction(SKAction.repeatAction(action, count: 3))

shape.runAction(SKAction.repeatActionForever(action)

shape.runAction(SKAction.sequence([action1,action2,action3]));
```

Combining Actions




```
node.runAction(SKAction.sequence([action1,action2,action3]))
```



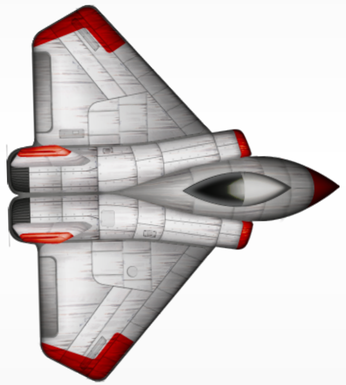

```
node.runAction(SKAction.group([action1,action2,action3]))
```

Other Actions

Texture animate

```
SKAction.animateWithTextures([action1,action2,action3] timePerFrame:0.1)
```

Path animate

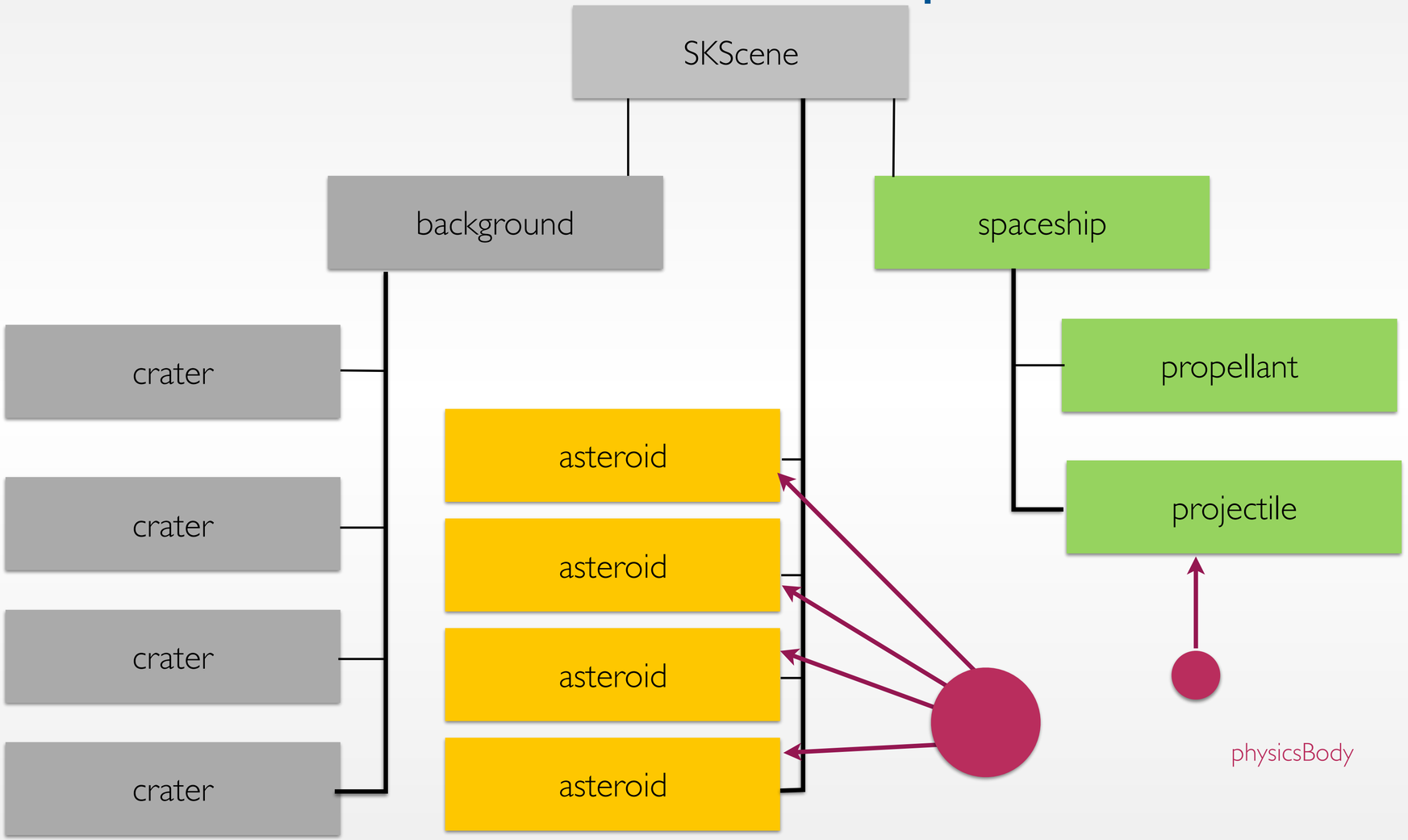


```
SKAction.followPath(aPath timePerFrame:2.5)
```

and many more: colors, sounds, custom blocks ...

Sprite Kit: Physics

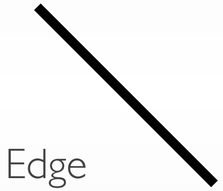
Scene Graph



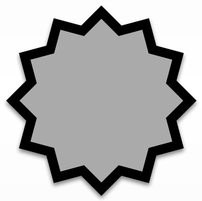
SKPhysicsBody

```
asteroid.physicsBody = SKPhysicsBody(circleOfRadius: 2.0)

asteroid.physicsBody = SKPhysicsBody(texture: sprite.texture, size: sprite.size)
```

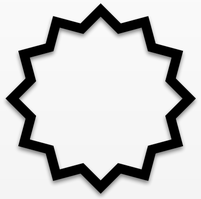
EdgeLoopFromRect



Edge



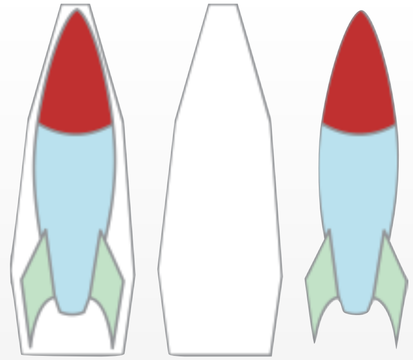
Polygon



EdgeLoopFromPath

EdgeChain

Rectangle



SKPhysicsWorld

- Each scene as its own PhysicsWorld
- Performs contact and collision tests

```
/* normal gravity */
self.physicsWorld.gravity = CGPointMake(0.0, -9.8);

/* inverted gravity */
self.physicsWorld.gravity = CGPointMake(0.0, +9.8);
```

SKPhysicsContact

Contact Delegate

```
self.physicsWorld.contactDelegate = self
```

projectile

asteroid

```
func didBeginContact(_ contact: SKPhysicsContact)
```

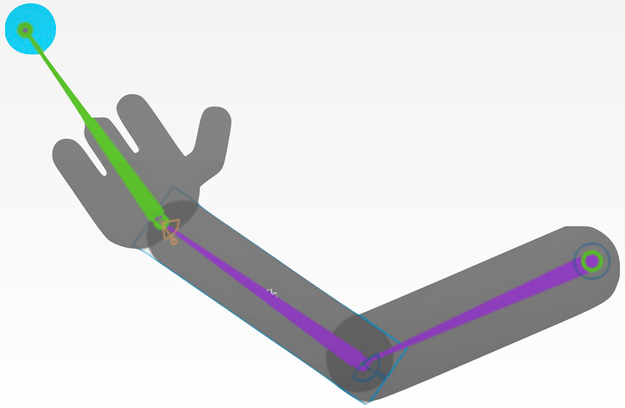
```
var bodyA: SKPhysicsBody { get }

var bodyB: SKPhysicsBody { get }

var contactPoint: CGPoint { get }

var collisionImpulse: CGFloat { get }

var contactNormal: CGVector { get }
```

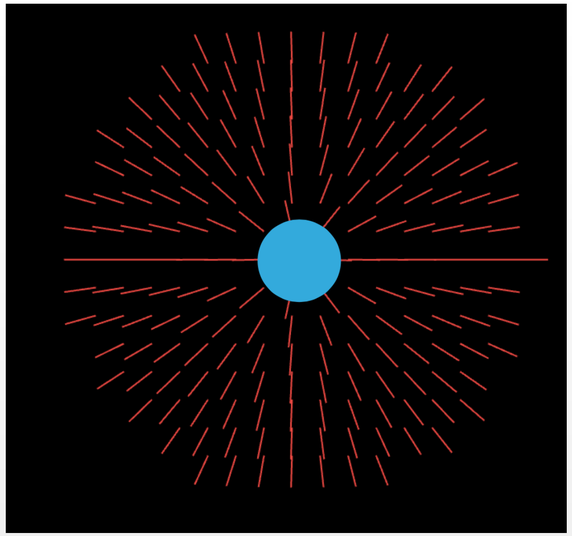
Physics Groups

```
  /**
     Defines what logical 'categories' this body belongs to. Defaults to all
bits set (all categories).
     */
    public var categoryBitMask: UInt32

    /**
     Defines what logical 'categories' of bodies this body responds to
collisions with. Defaults to all bits set (all categories).
     */
    public var collisionBitMask: UInt32

    /**
     Defines what logical 'categories' of bodies this body generates
intersection notifications with. Defaults to all bits cleared (no categories).
     */
    public var contactTestBitMask: UInt32
```

More Physics

- Inverse Kinematics
- Physics Fields
  - Fields simulate physical forces
  - Linear gravity field
  - Radial gravity field
  - Electric field

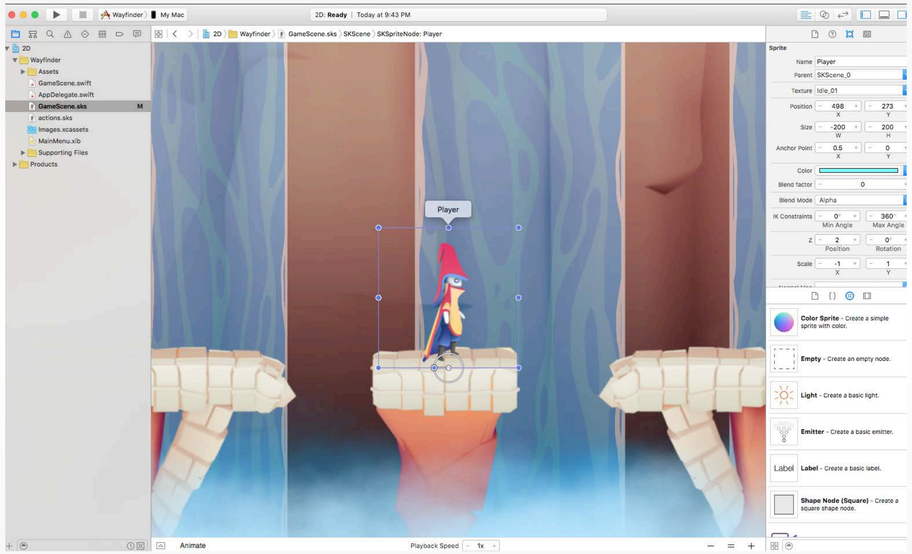


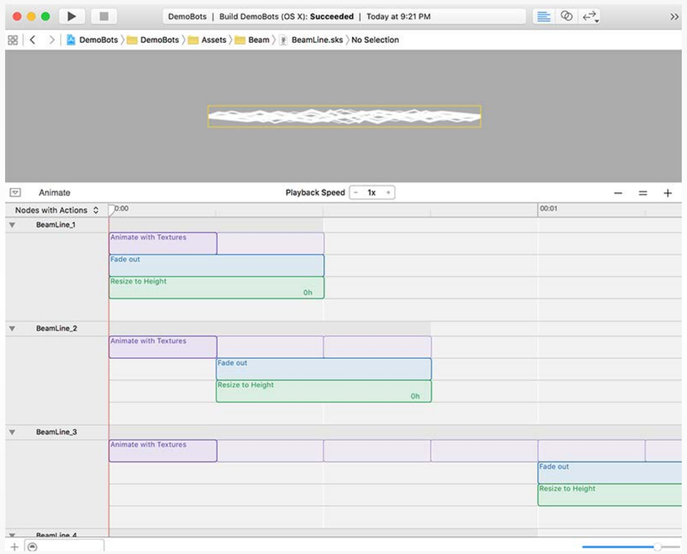


Physics Field Demo

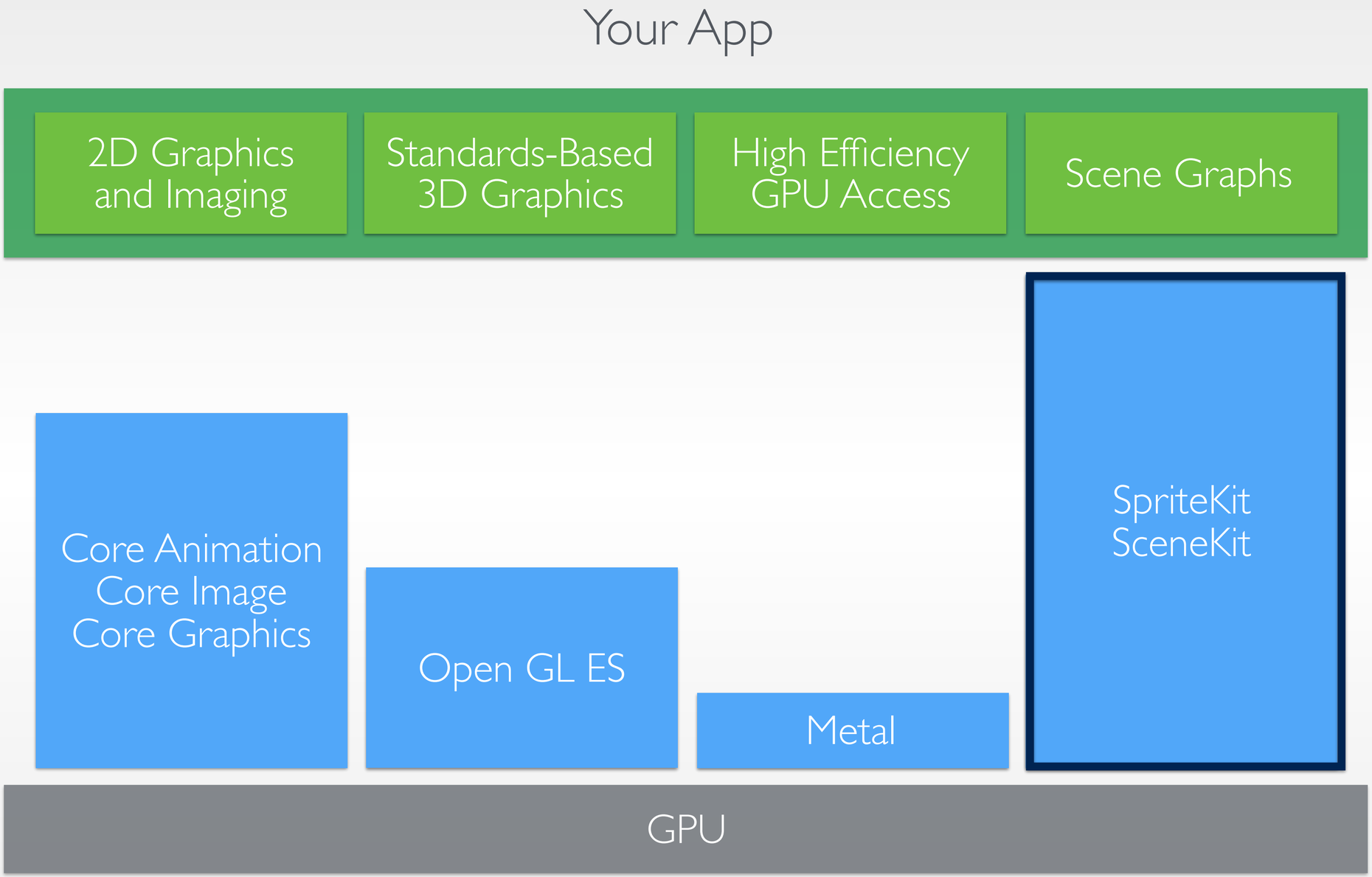
Sprite Kit Editor

- 2D editor in Xcode
- Visual layout
- Animation editor
- Physics simulation





Apple (2015)

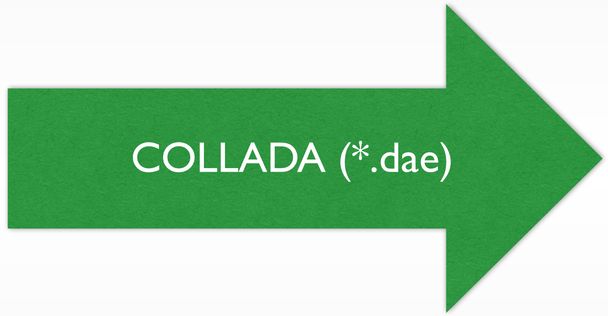


Apple, WWDC 2014

Scene Kit

Apple (2013)

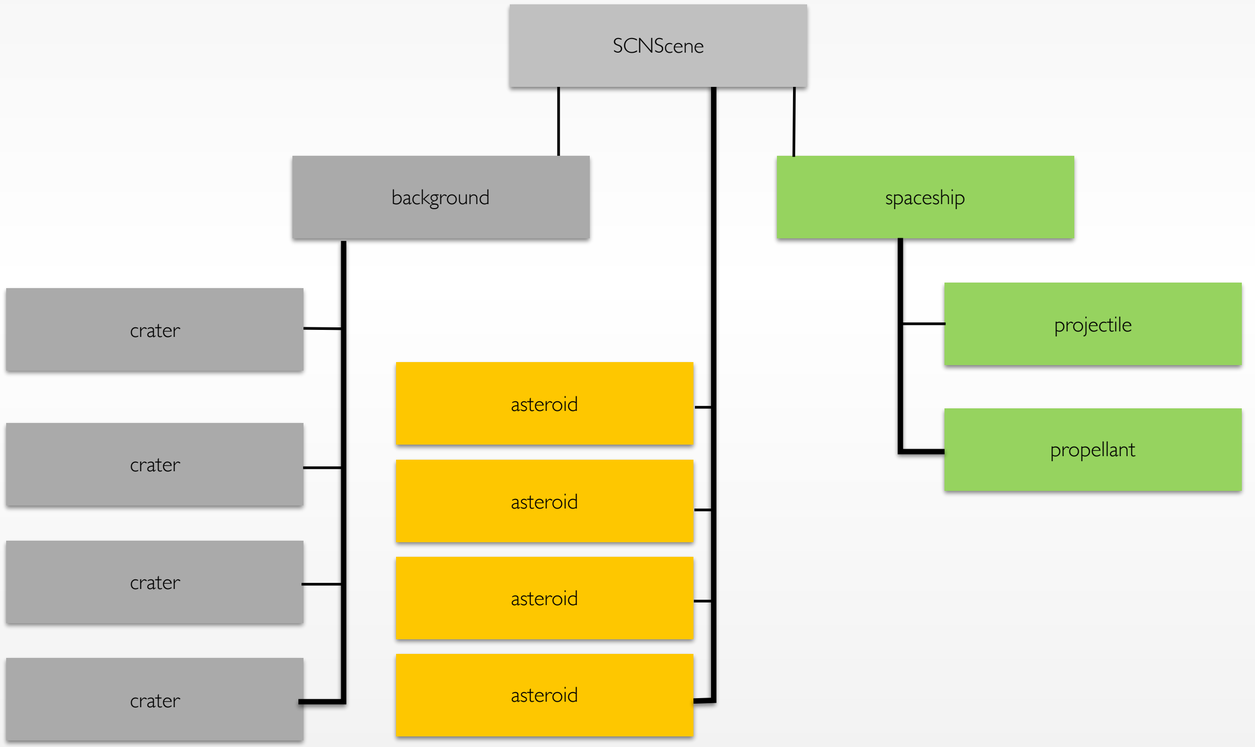
Scene Kit



Scene Kit

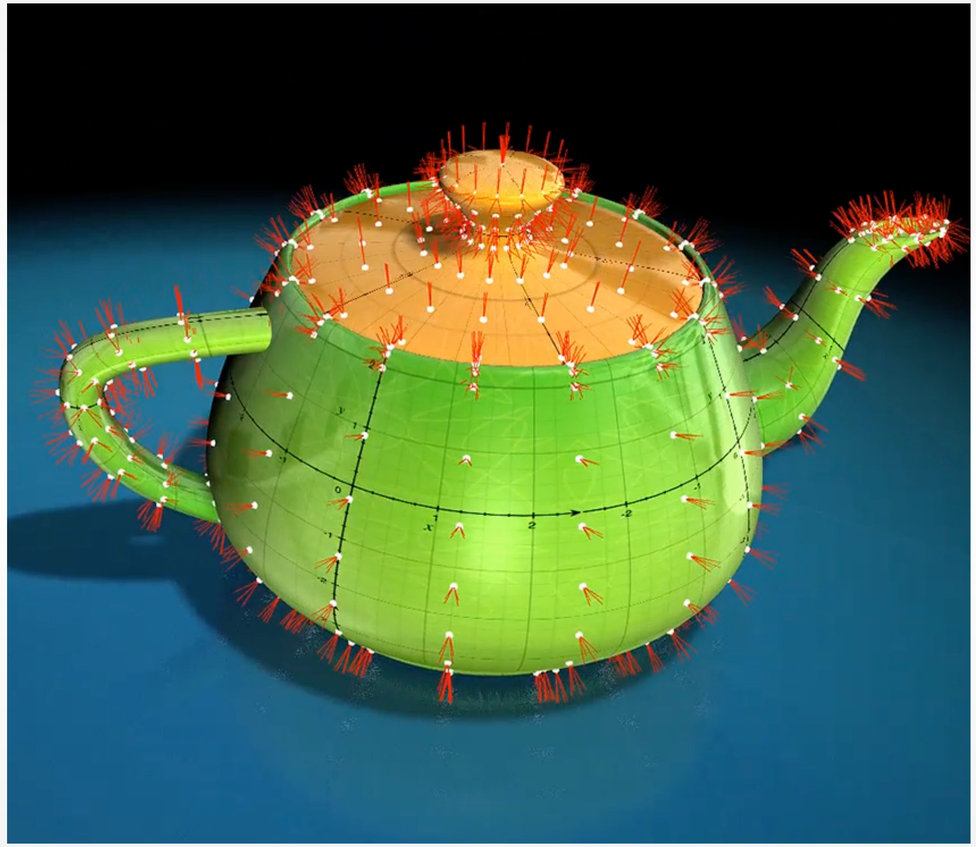
Scene Kit

- Similar to Sprite Kit but in 3D
- Scene class: SCNScene
- Root class SCNNode
- Scene Graph
  - Geometry
  - Light
  - Camera



SCNNode

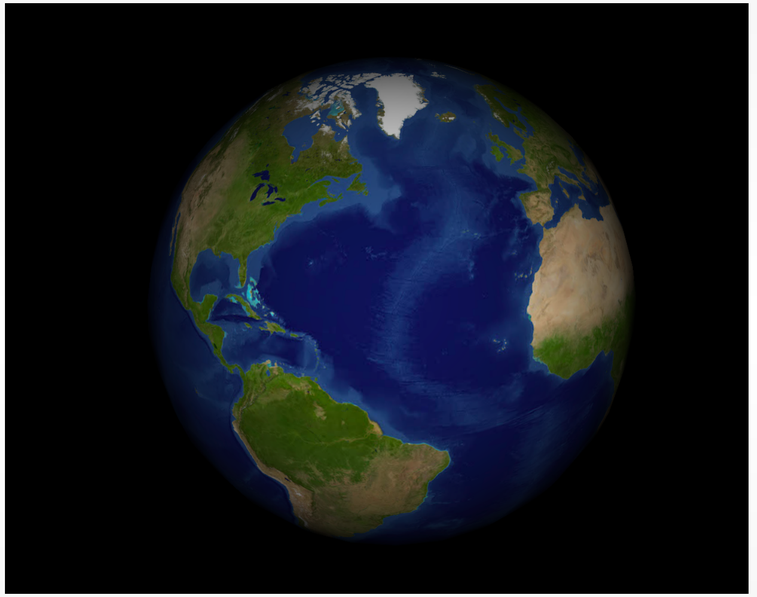
- SCNGeometry
  - Triangles
  - Vertices
  - Normals
  - UVs
  - Materials
- SCNCamera



Apple WWDC 2014

SCNMaterials

- Determines the appearance of the geometry
- SCNMaterialProperty
- Contains colors or images
- Lighting and shading attributes
  - Defuse
  - Ambient
  - Specular
  - Normal
  - …



Animations and Actions

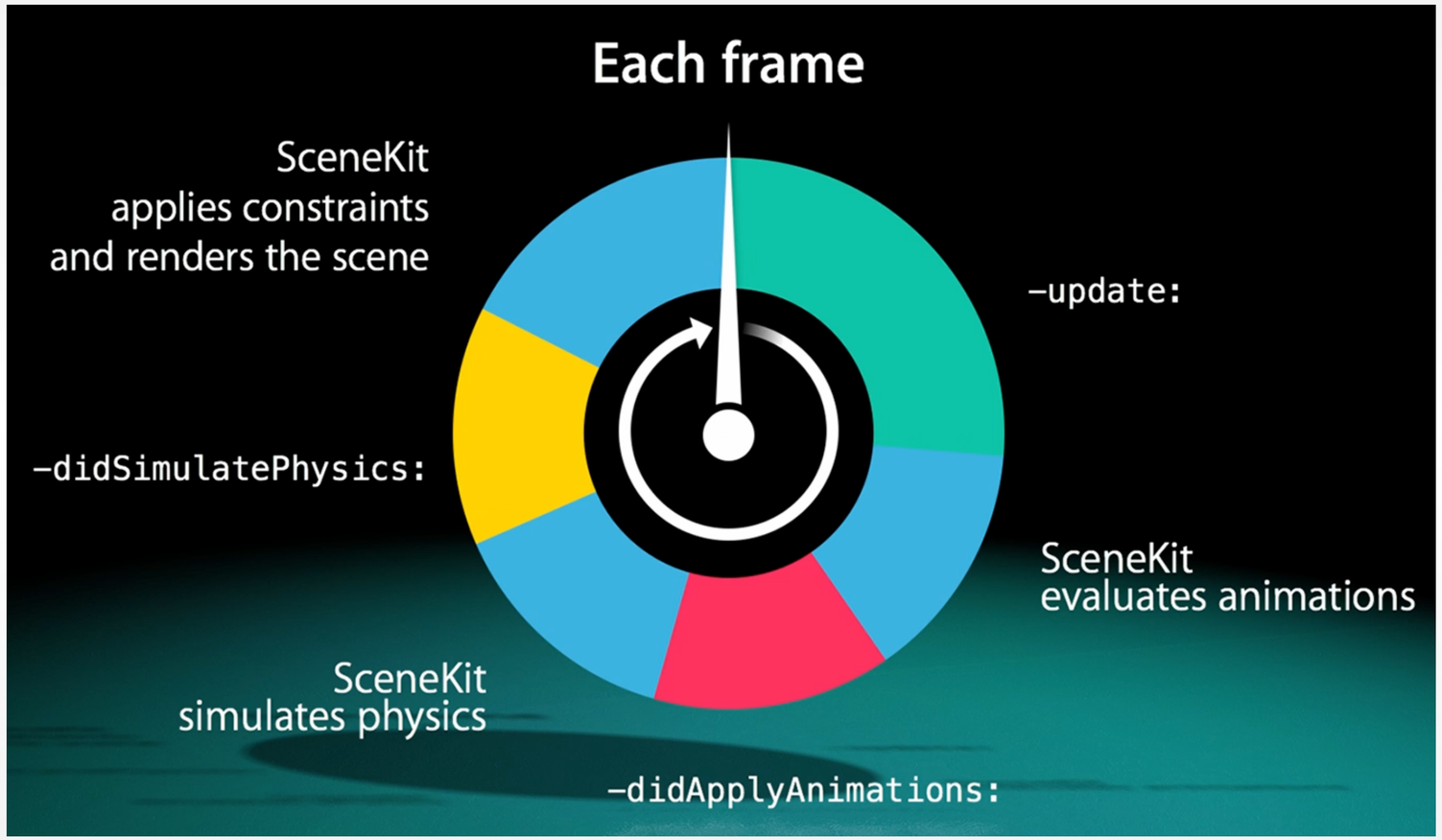
Transaction like Core Animation

```
// highlight it
            SCNTransaction.begin()
            SCNTransaction.setAnimationDuration(0.5)

            // on completion - unhighlight
            SCNTransaction.setCompletionBlock {
                SCNTransaction.begin()
                SCNTransaction.setAnimationDuration(0.5)

                material.emission.contents = UIColor.blackColor()

                SCNTransaction.commit()
            }
```

Action like in Sprite Kit (SCNAction)

```
SCNAction.rotateByAngle(M_PI, aroundAxis: SCNVector3Make(0, 1, 0), duration: 5)
```

Scene Kit Render Loop



Apple WWDC 2014

Physics

- Similar to Sprite Kit

```
aNode.physicsBody?.applyForce(aVector3, atPosition: aVector3, impulse: true)

aNode.physicsBody?.physicsShape = SCNPhysicsShape(aGeometry,options)
```

- Inverse Kinematics
- Constraints (SCNConstraints)
- Force fields

Scene Kit and Sprite Kit

- Include Sprite Kit in Scene Kit

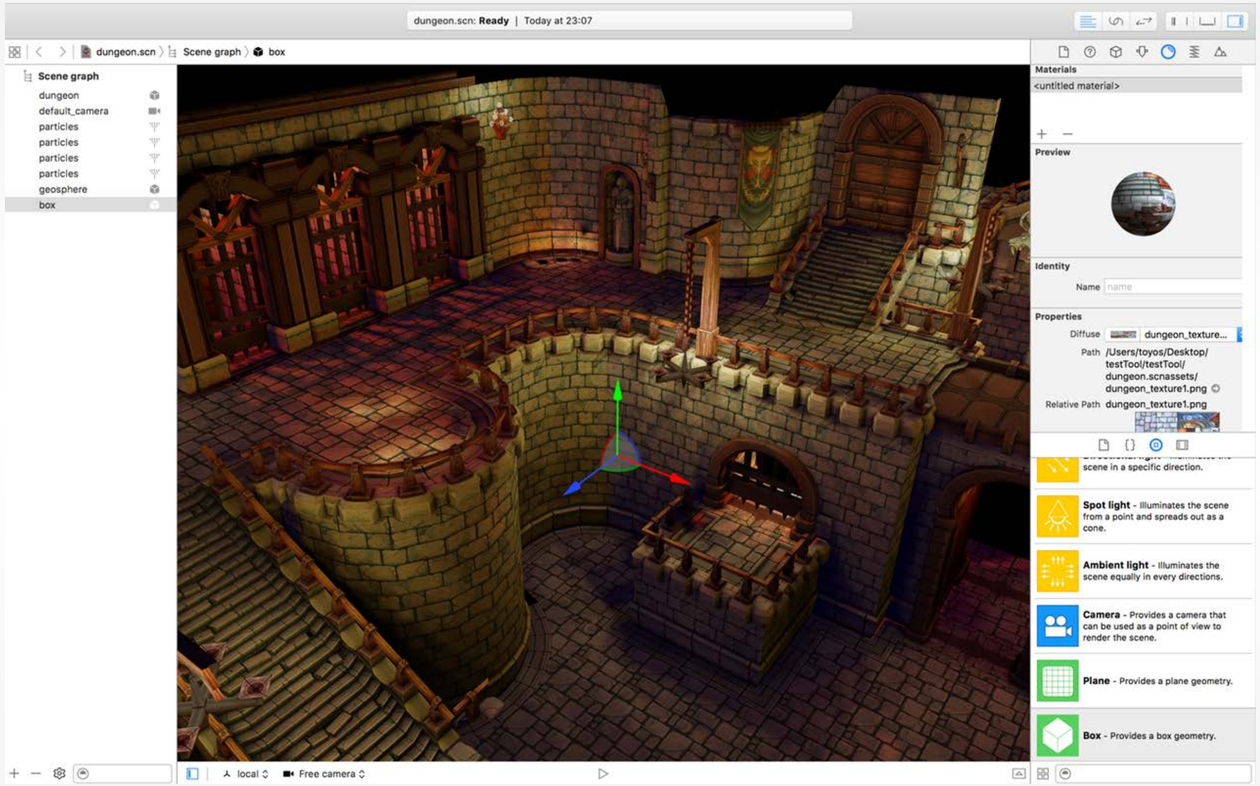
```
let spriteScene = SKScene(size: aSize)
let material = SCNMaterial()

material.diffuse.contents = spriteScene
```

- Include Scene Kit in Sprite Kit

```
let node = SK3DNode(WithViewportSize:aPort)
node.scnScene = node
```

Scene Editor

- 3D scene editor in Xcode
- Visual layout
- Animation editor
- Physics simulation



Apple (2015)

Scene Kit Demo

Summary

- OpenGL (WS 2012/2013)
- Metal (WWDC 2014/2015)
  - Powerful low level API
- Sprite Kit
  - Easy to use 2D game engine
- Scene Kit
  - Easy to use 3D game engine