Introduction to Sprite Kit

Session 502

Jacques Gasselin de Richebourg
Tim Oriol

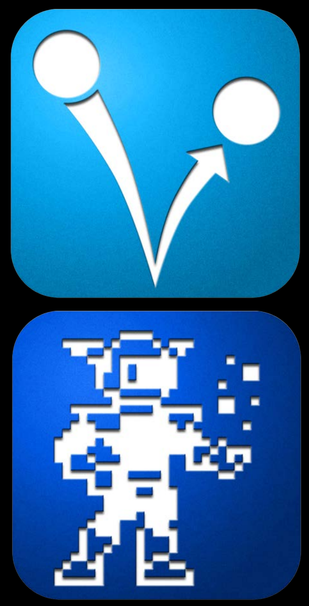
Background

- Incredible variety of games on the store
  - Iconic games—Many are 2D
- Developers have common needs
  - Lots of beautiful graphics—Fast
    - Particles and visual effects
    - Physics and animation
- Focus on developing games instead of engines

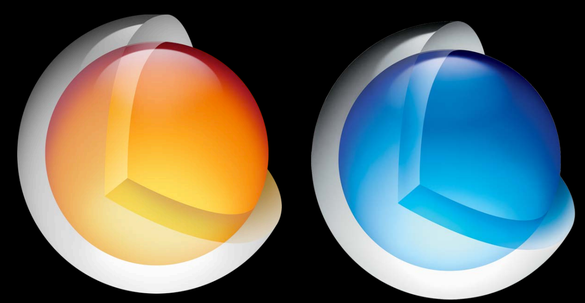
Sprite Kit

Enhancing 2D game development

Images of Sprites, Shapes and Particles



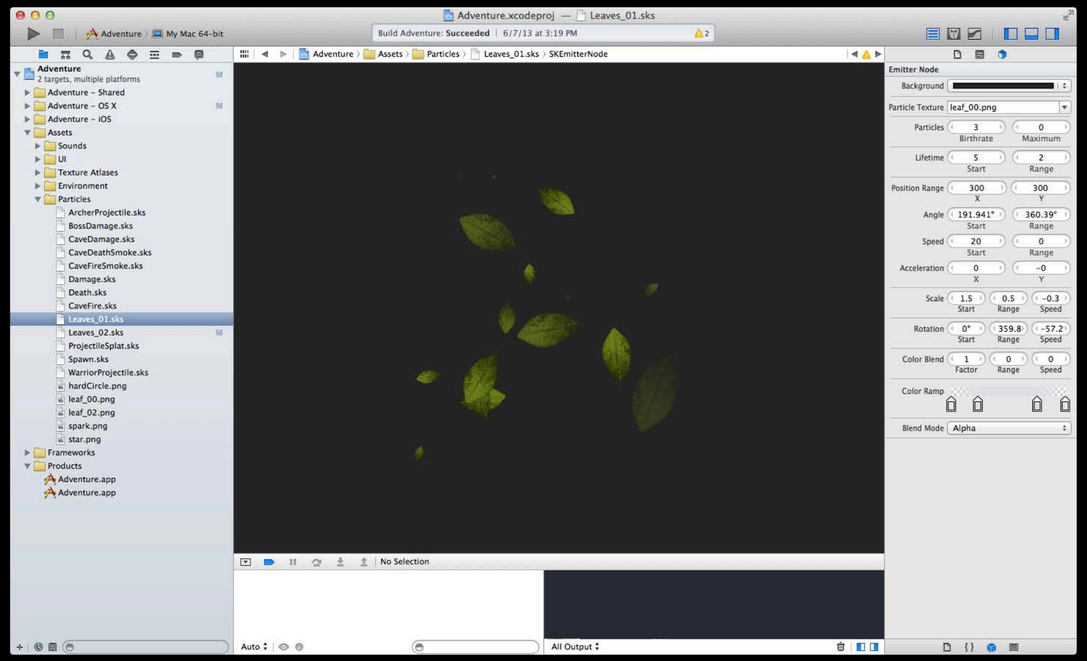
Animations and Physics

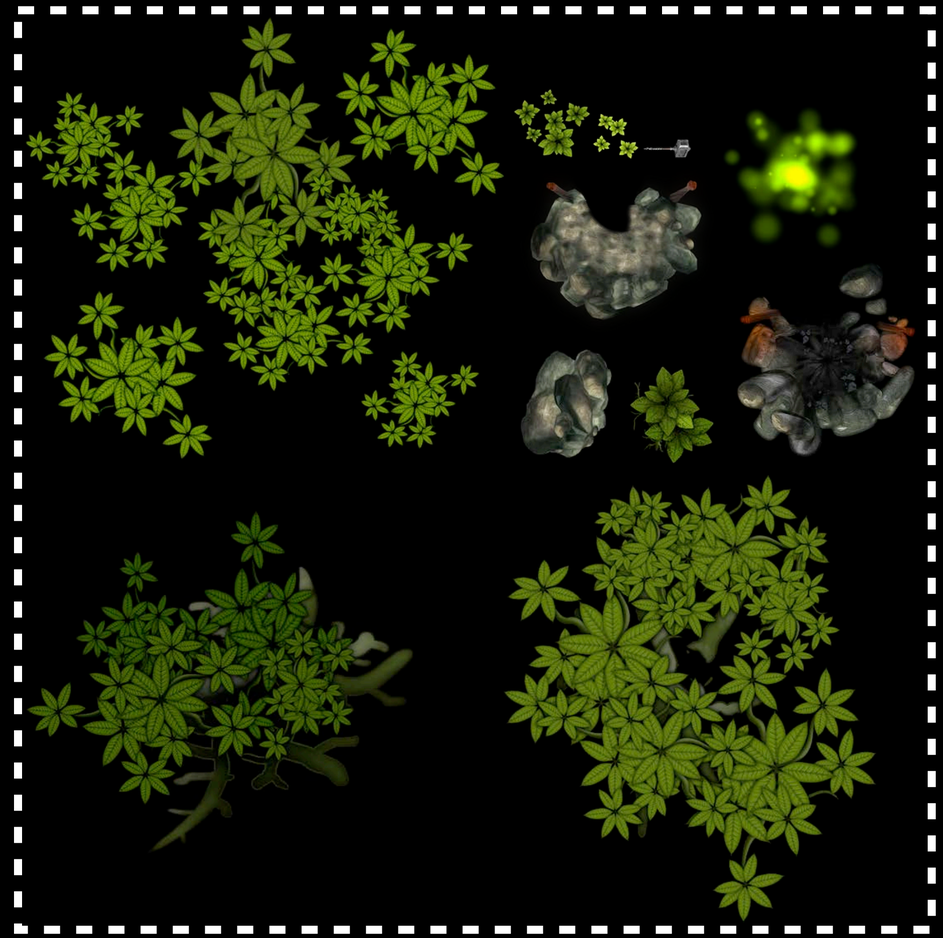


Audio, Video, Visual Effects

Sprite Kit

Enhancing 2D game development





Agenda

- Introduction to Sprite Kit
  - Node types
  - Effects and actions
  - Physics
- Designing games using Sprite Kit
  - Demo ‘take-home’ sample with complete documentation
  - Managing the art pipeline—Creating, editing and using art
  - Detailed look at Xcode support

Demo

Adventure

Sprite Kit Basics

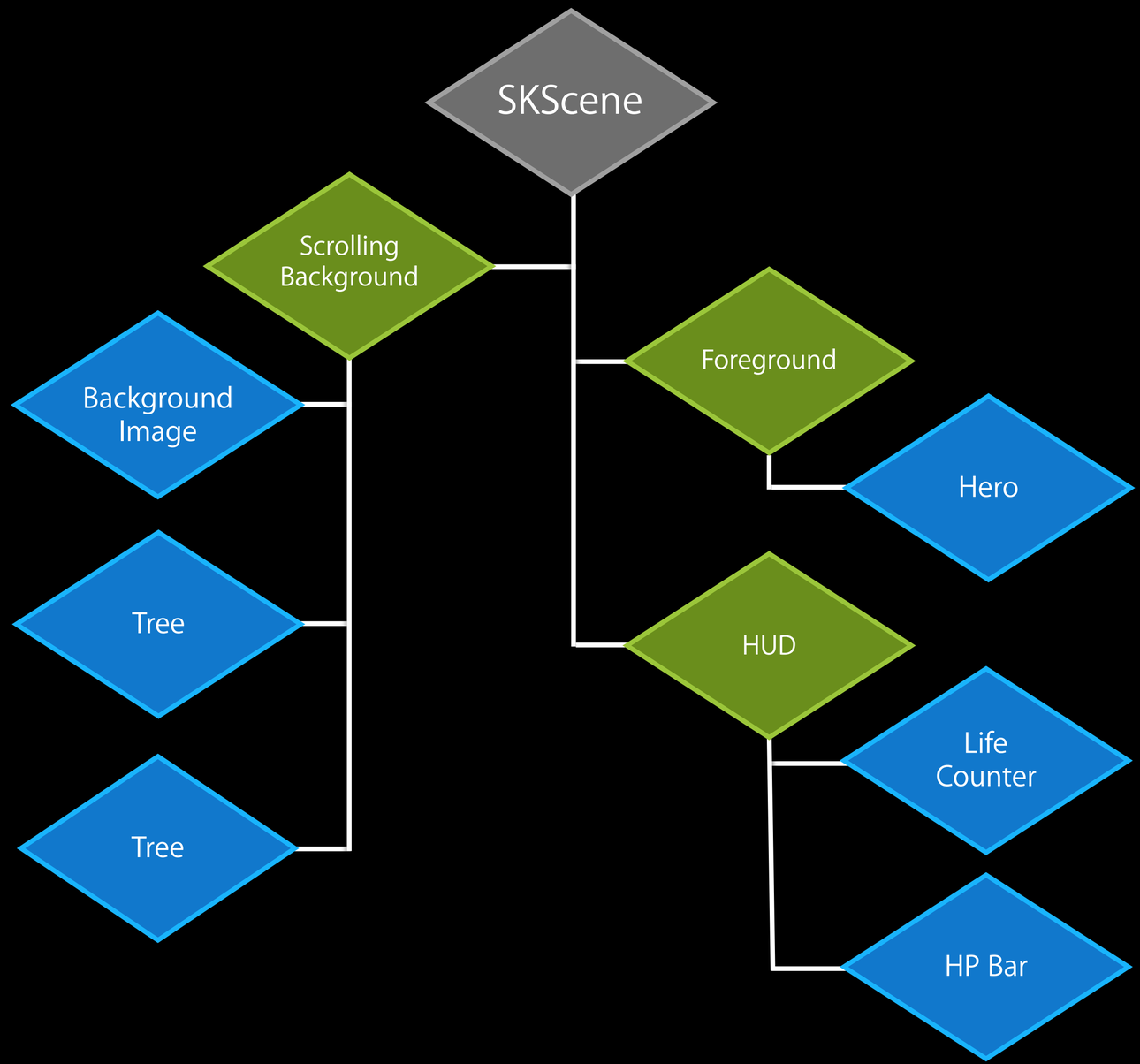
The Parts of a Sprite Kit Game

Scenes

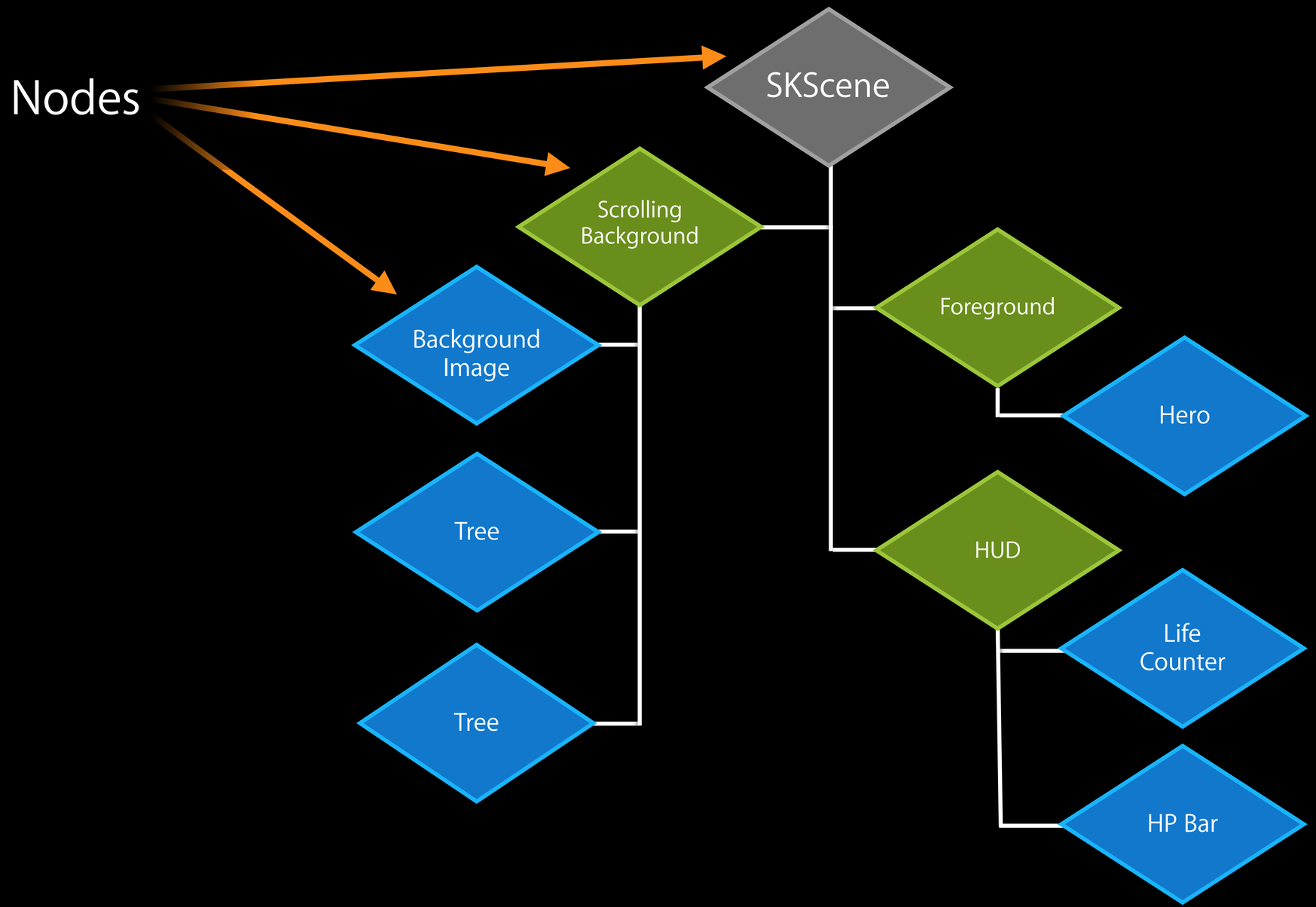
Actions

Physics

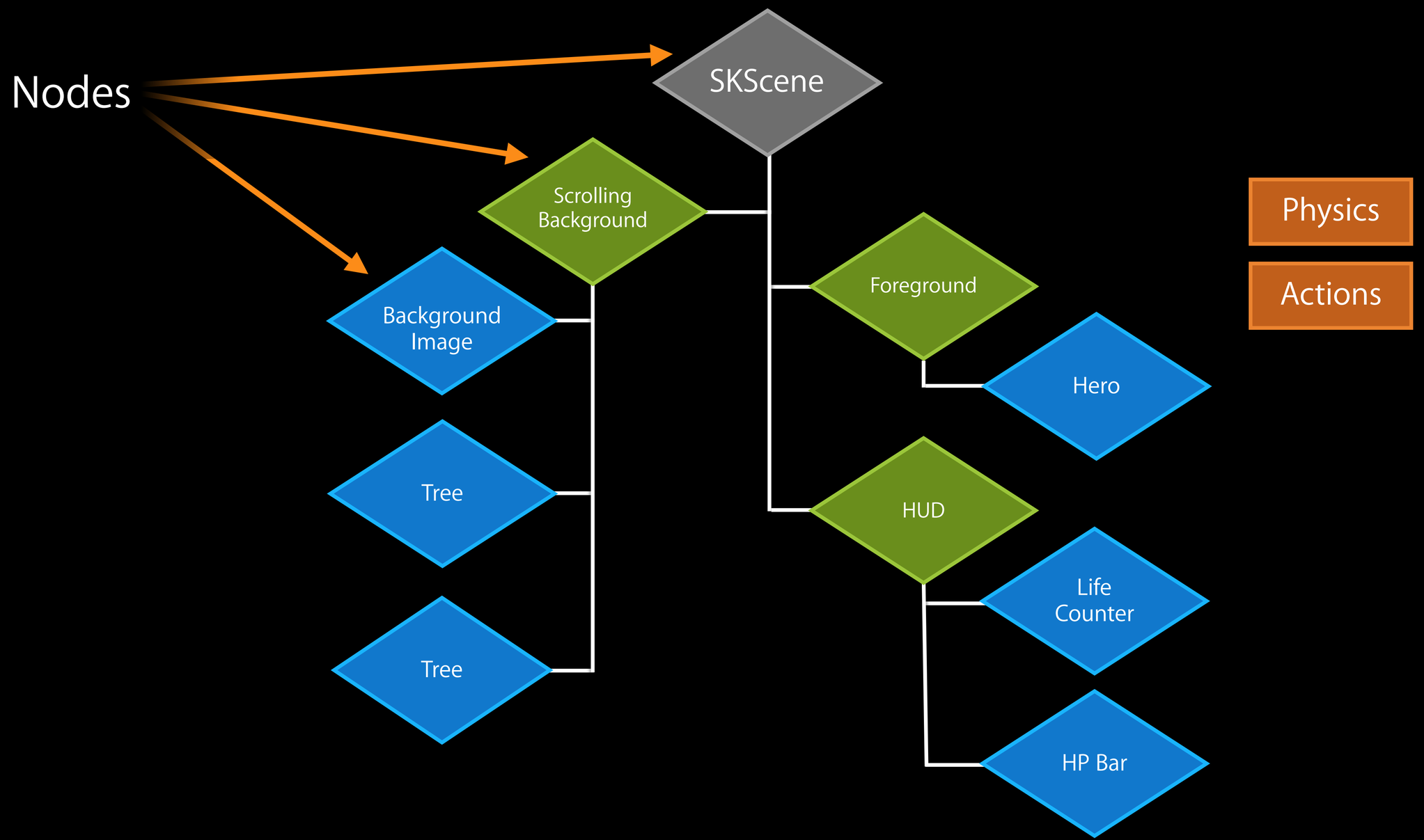
Scene Graph


SKScene
Scrolling Background
Background Image
Tree
Tree
Foreground
Hero
HUD
Life Counter
HP Bar

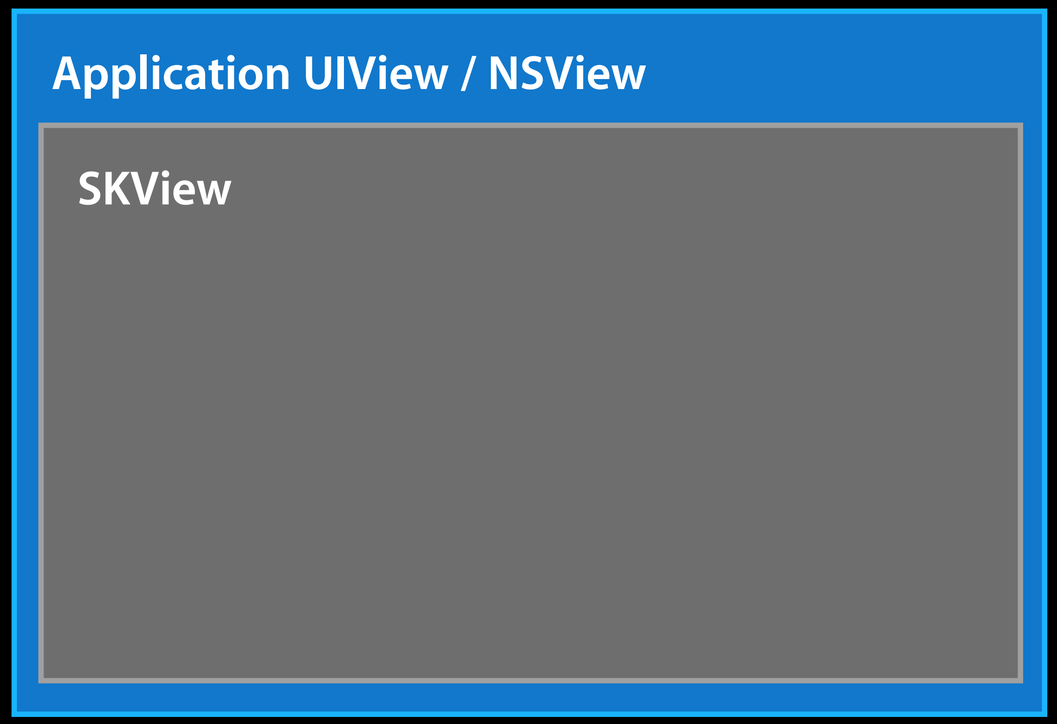
Scene Graph


Nodes
SKScene
Scrolling Background
Background Image
Tree
Tree
Foreground
Hero
HUD
Life Counter
HP Bar

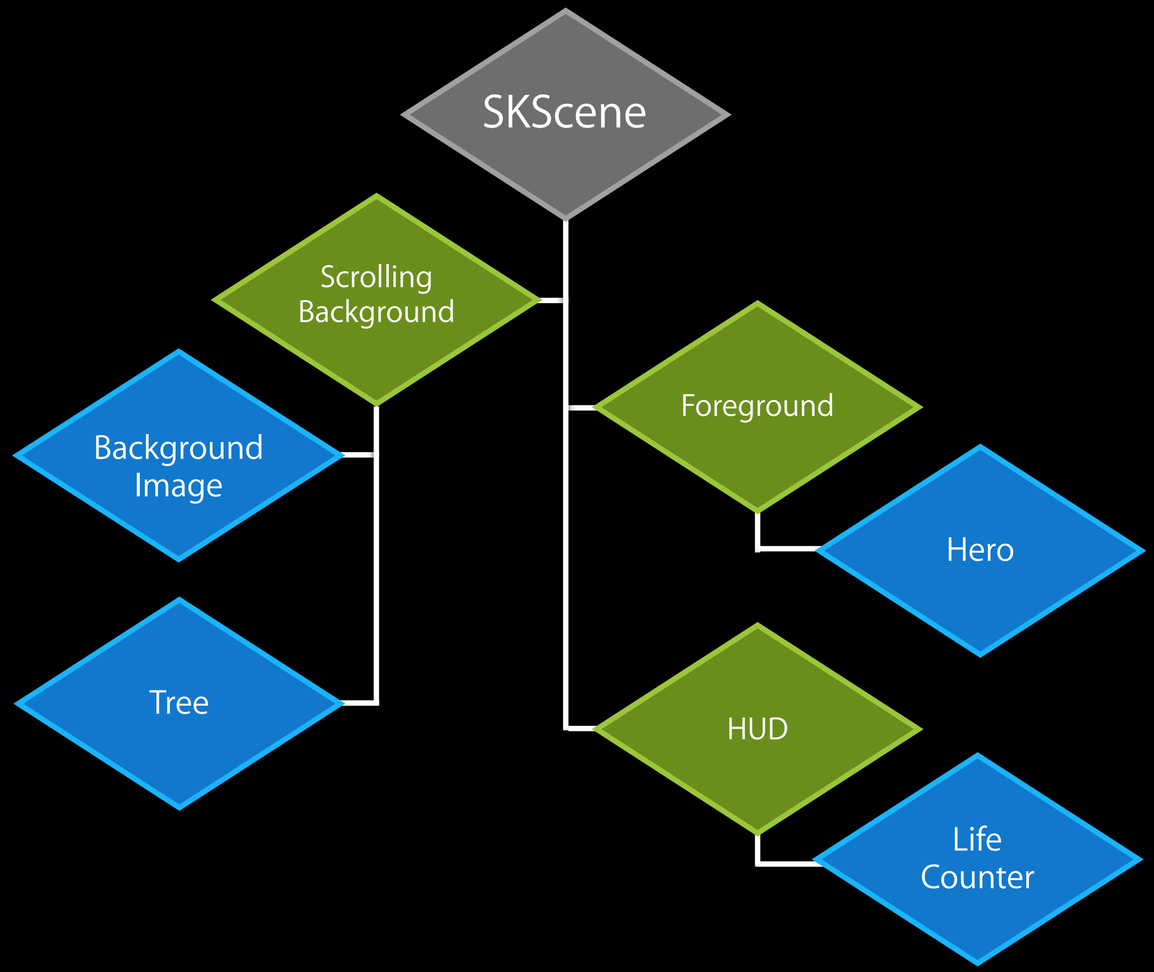
Scene Graph


Nodes
SKScene
Scrolling Background
Background Image
Tree
Tree
Foreground
Hero
HUD
Life Counter
HP Bar
Physics
Actions

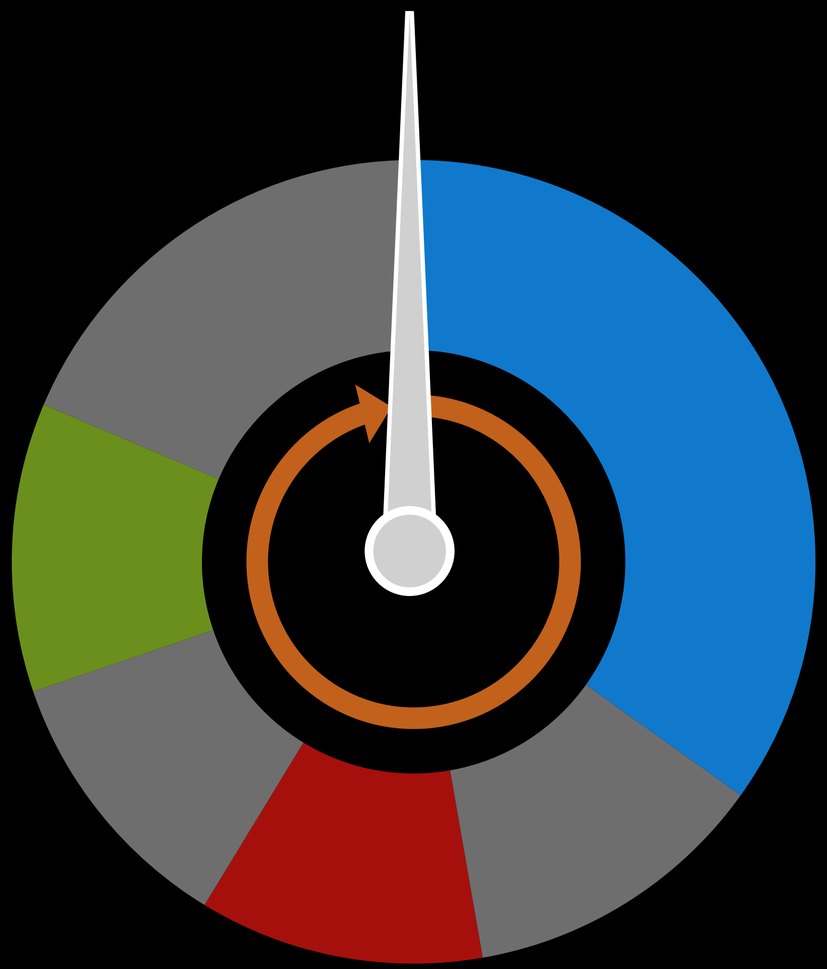
Displaying Sprite Kit Content



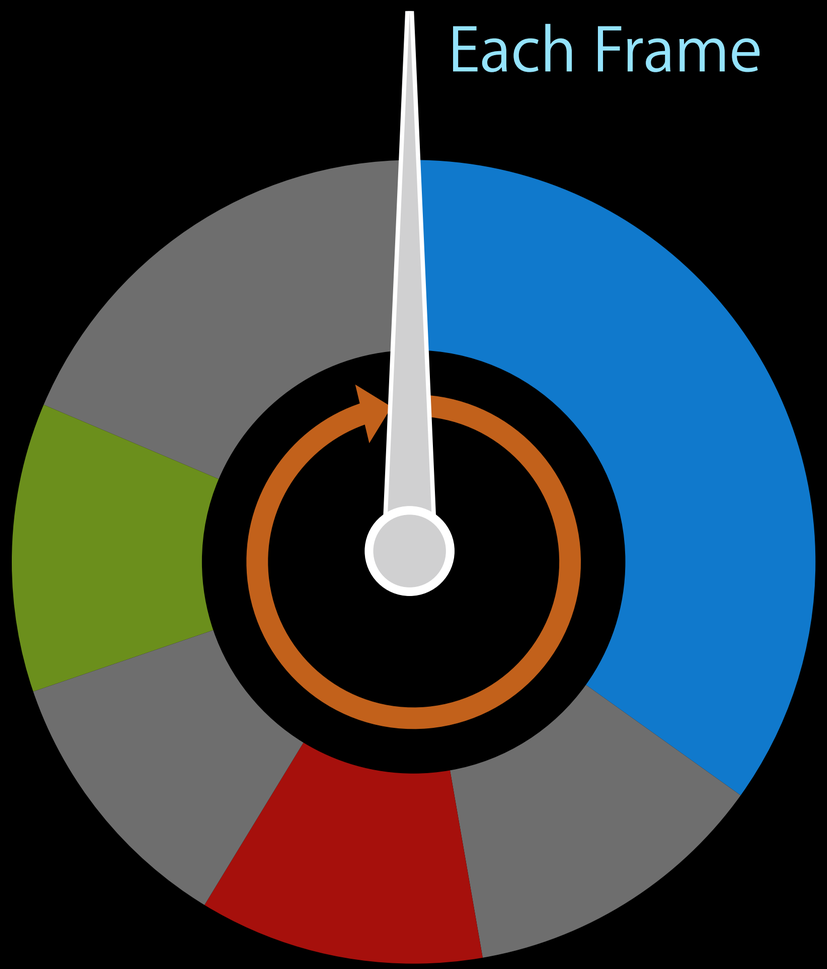

```
[skView presentScene: myScene];
```



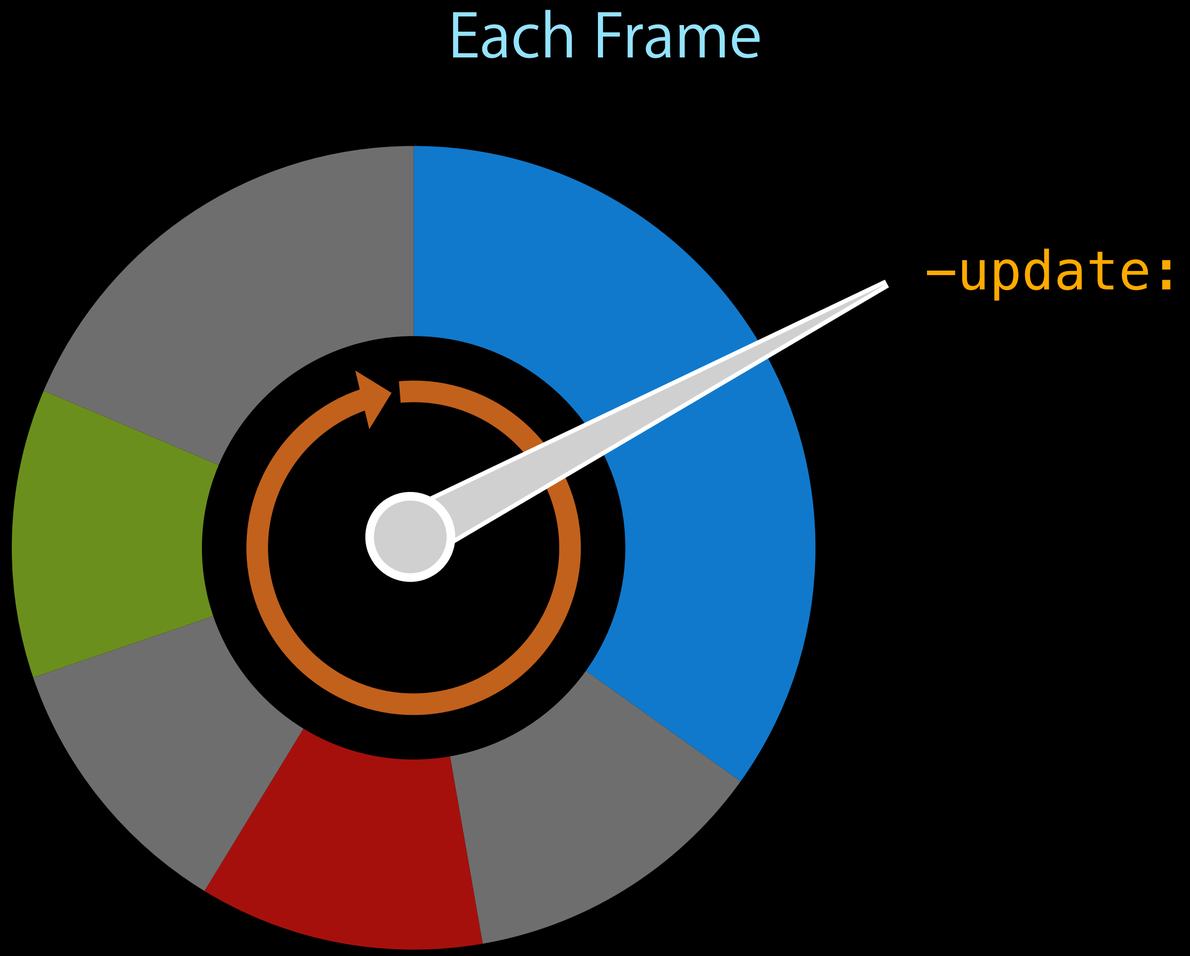
The Sprite Kit Game Loop



The Sprite Kit Game Loop

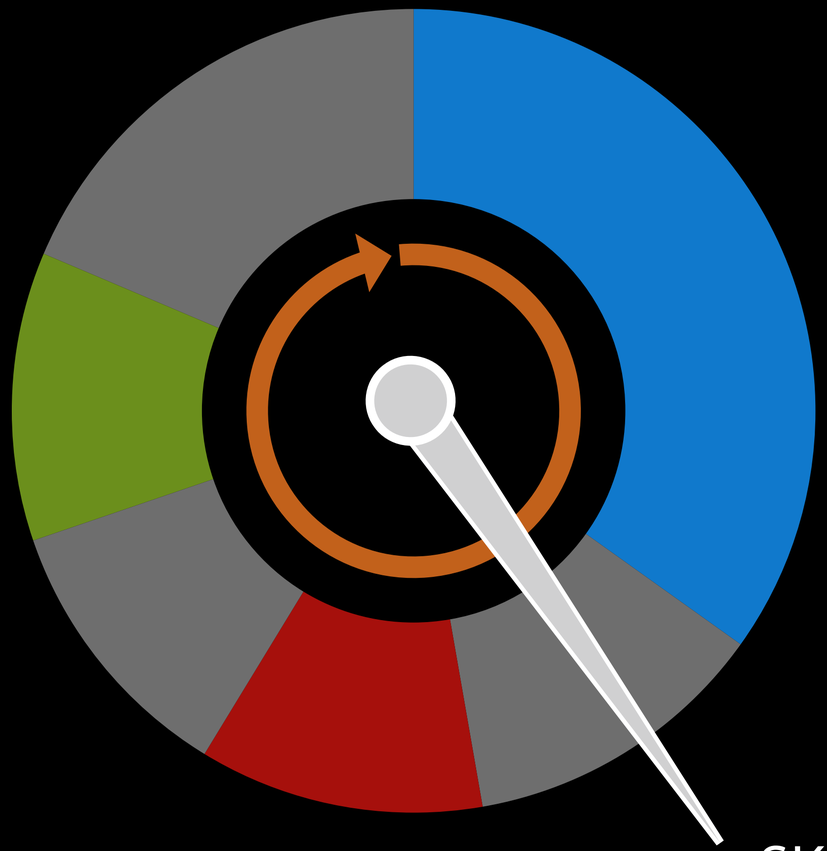


The Sprite Kit Game Loop



The Sprite Kit Game Loop

Each Frame

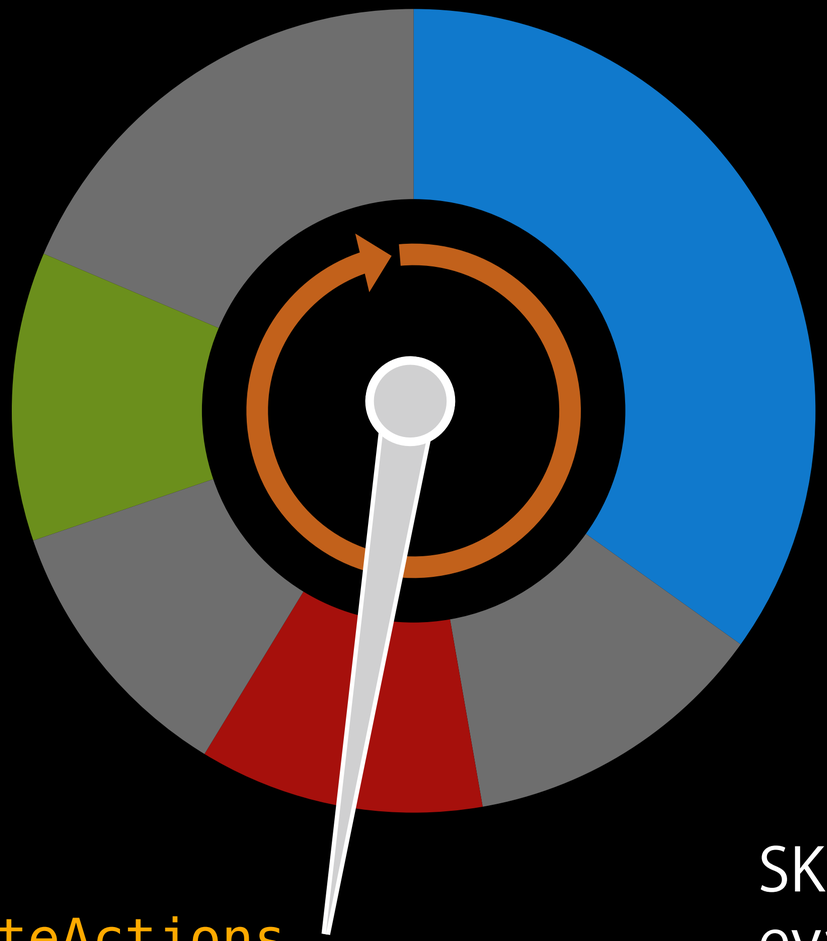


`-update:`

SKScene
evaluates actions

The Sprite Kit Game Loop

Each Frame



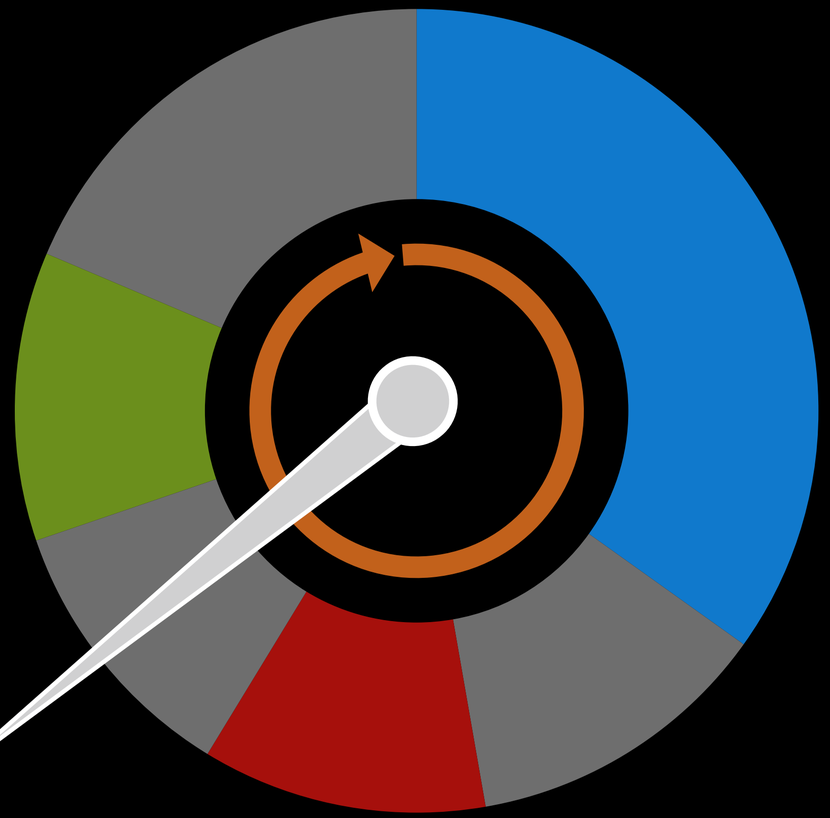
`-update:`

`-didEvaluateActions`

SKScene
evaluates actions

The Sprite Kit Game Loop

Each Frame



-update:

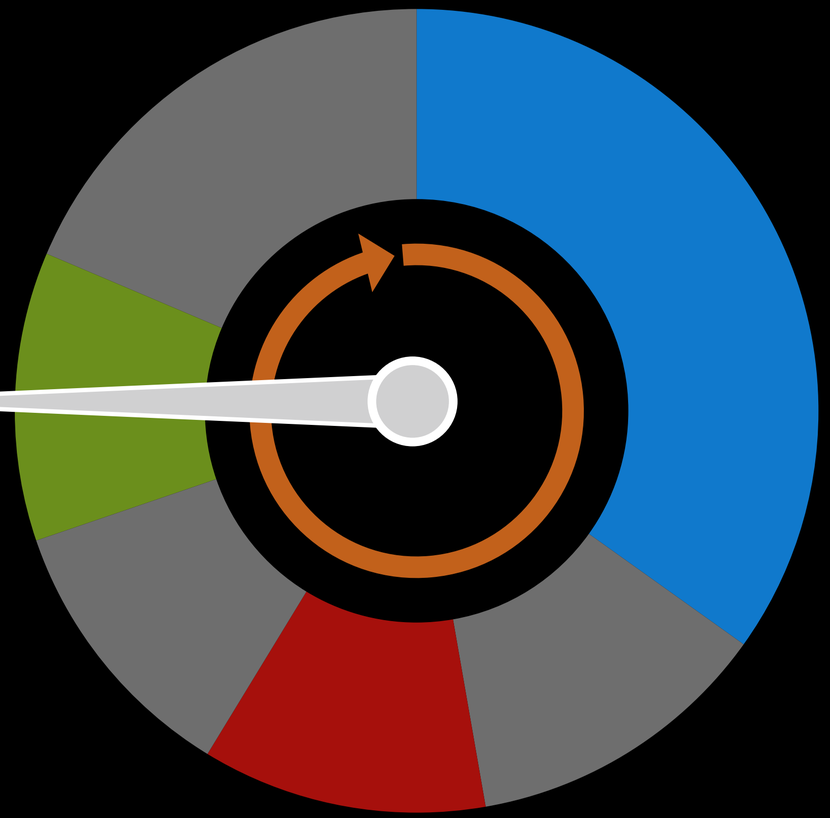
SKScene
simulates physics

-didEvaluateActions

SKScene
evaluates actions

The Sprite Kit Game Loop

Each Frame



-update:

-didSimulatePhysics

SKScene
simulates physics

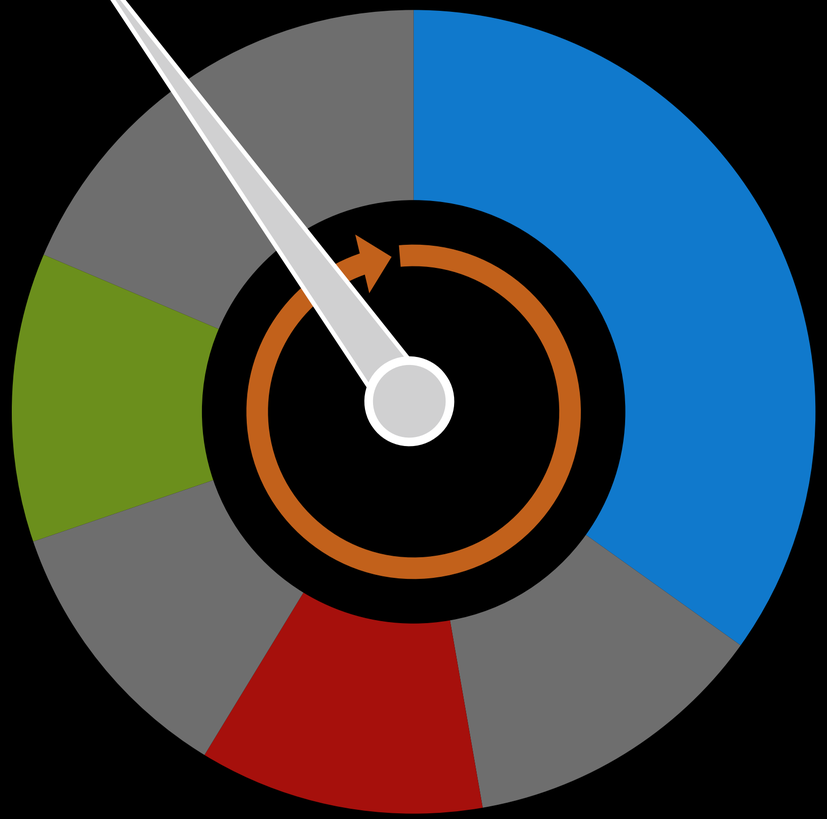
-didEvaluateActions

SKScene
evaluates actions

The Sprite Kit Game Loop

SKView
renders the scene

Each Frame



-update:

-didSimulatePhysics

SKScene
simulates physics

-didEvaluateActions

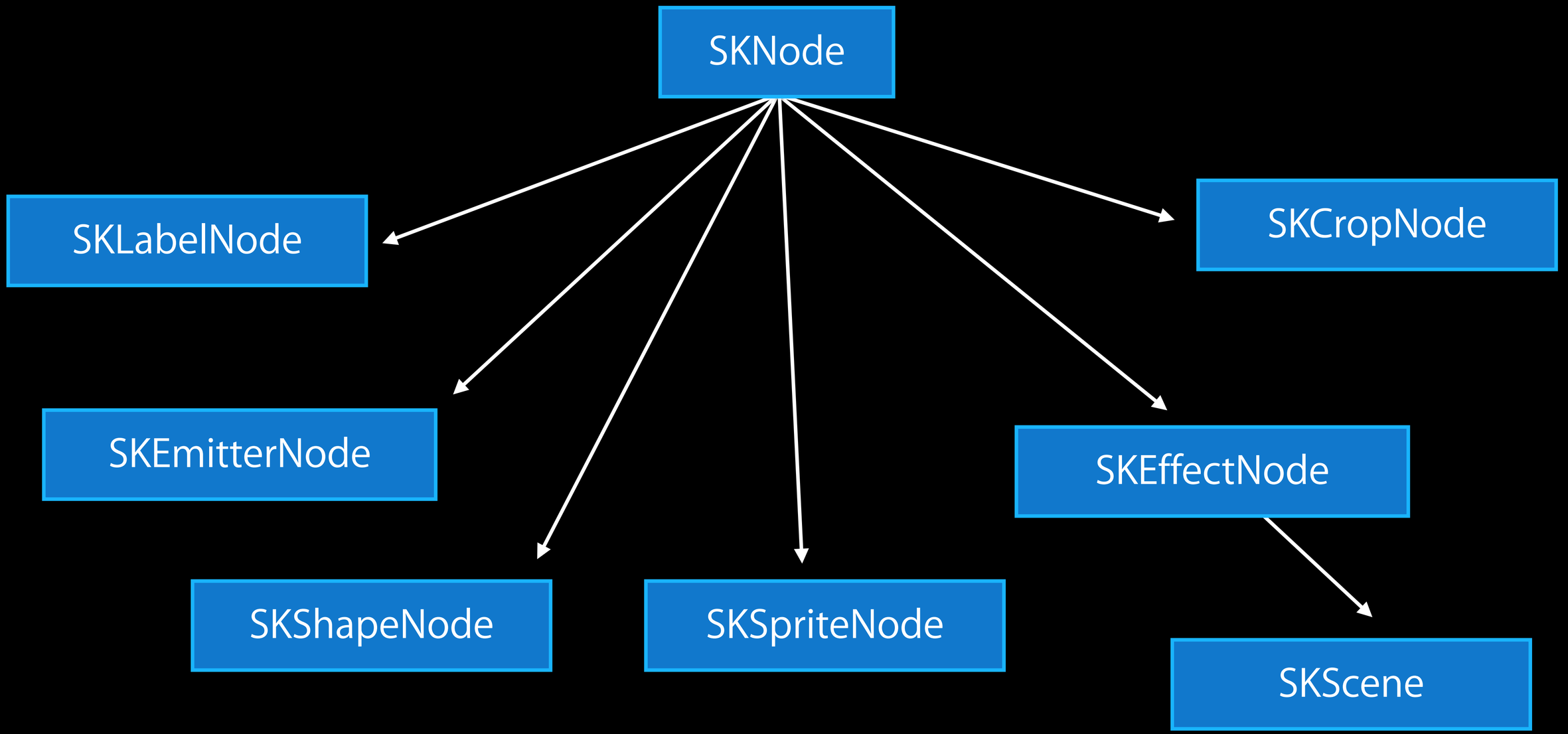
SKScene
evaluates actions

Demo

Sprite Kit template

Sprite Kit Nodes

Sprite Kit Nodes

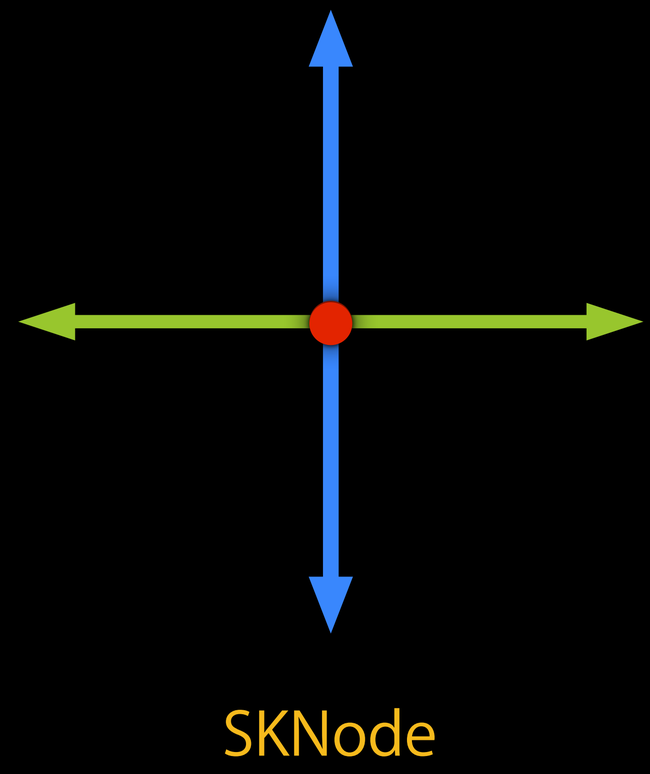


SKNode

The basic node

- Used for grouping or a handle to transform children

```
/* Position in the parent's coordinate space */
@property CGPoint position;

/* Rotation about the z axis (in radians) */
@property CGFloat zRotation;

/* The scaling along the X axis */
@property CGFloat xScale;

/* The scaling along the Y axis */
@property CGFloat yScale;

/* Alpha (multiplied by the output color) */
@property CGFloat alpha;

/* Hidden nodes (and their children) wont be rendered */
@property (getter = isHidden) BOOL hidden;
```



SKSpriteNode

Sprite Kit MVP

- Has a explicit size
- Can display a color
- Can display a texture

SKSpriteNode

Sprite Kit MVP

- Has a explicit size
- Can display a color
- Can display a texture

SKSpriteNode

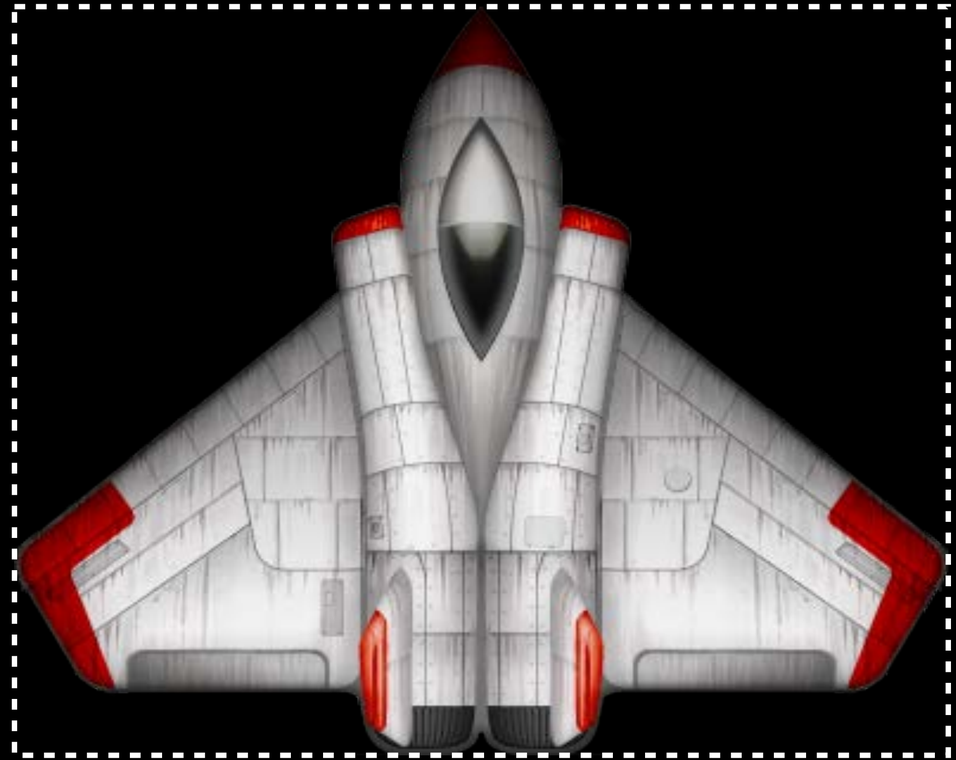
Sprite Kit MVP

- Has a explicit size
- Can display a color
- Can display a texture

SKSpriteNode

Sprite Kit MVP

- Has a explicit size
- Can display a color
- Can display a texture

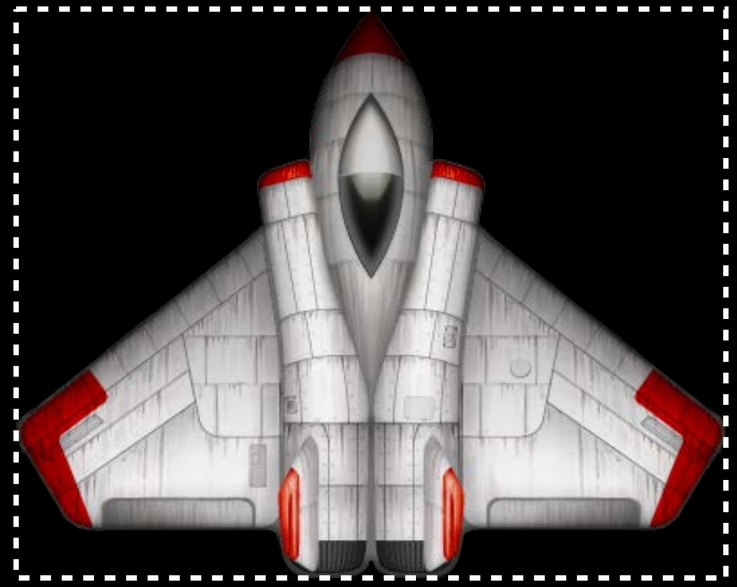


SKSpriteNode

SKTexture

Sprite Kit bitmap content

- Represents Sprite Kit bitmap data
- Automatically managed by the framework

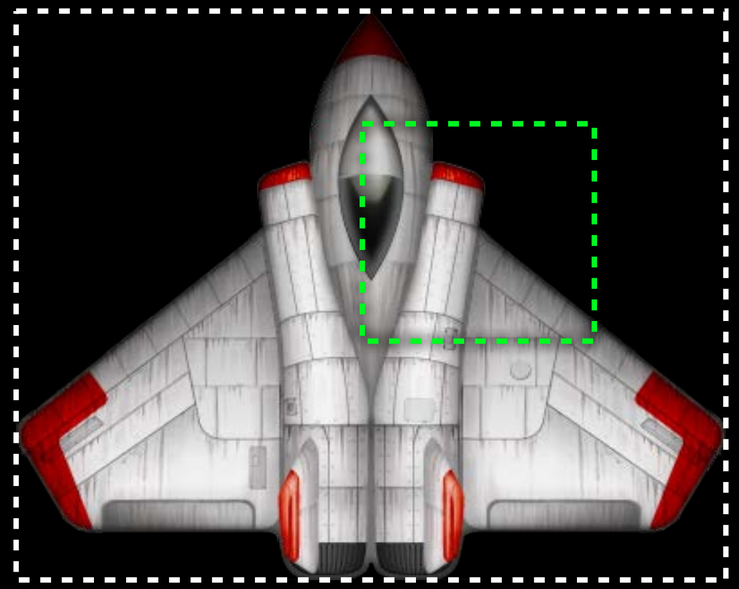


```
[SKTexture textureWithImageNamed:@"ship.png"];

[SKTexture textureWithCGImage:myCGImageRef];

[SKTexture textureWithData:rgbaNSData size:CGSizeMake(100, 100)];

[SKTexture textureWithImage:myUIImage];

[SKTexture textureWithRect:CGRectMake(100, 100, 80, 80) inTexture:tex1];
```

SKTexture

Sprite Kit bitmap content

- Represents Sprite Kit bitmap data
- Automatically managed by the framework



```
[SKTexture textureWithImageNamed:@"ship.png"];

[SKTexture textureWithCGImage:myCGImageRef];

[SKTexture textureWithData:rgbaNSData size:CGSizeMake(100, 100)];

[SKTexture textureWithImage:myUIImage];

[SKTexture textureWithRect:CGRectMake(100, 100, 80, 80) inTexture:tex1];
```

SKSpriteNode

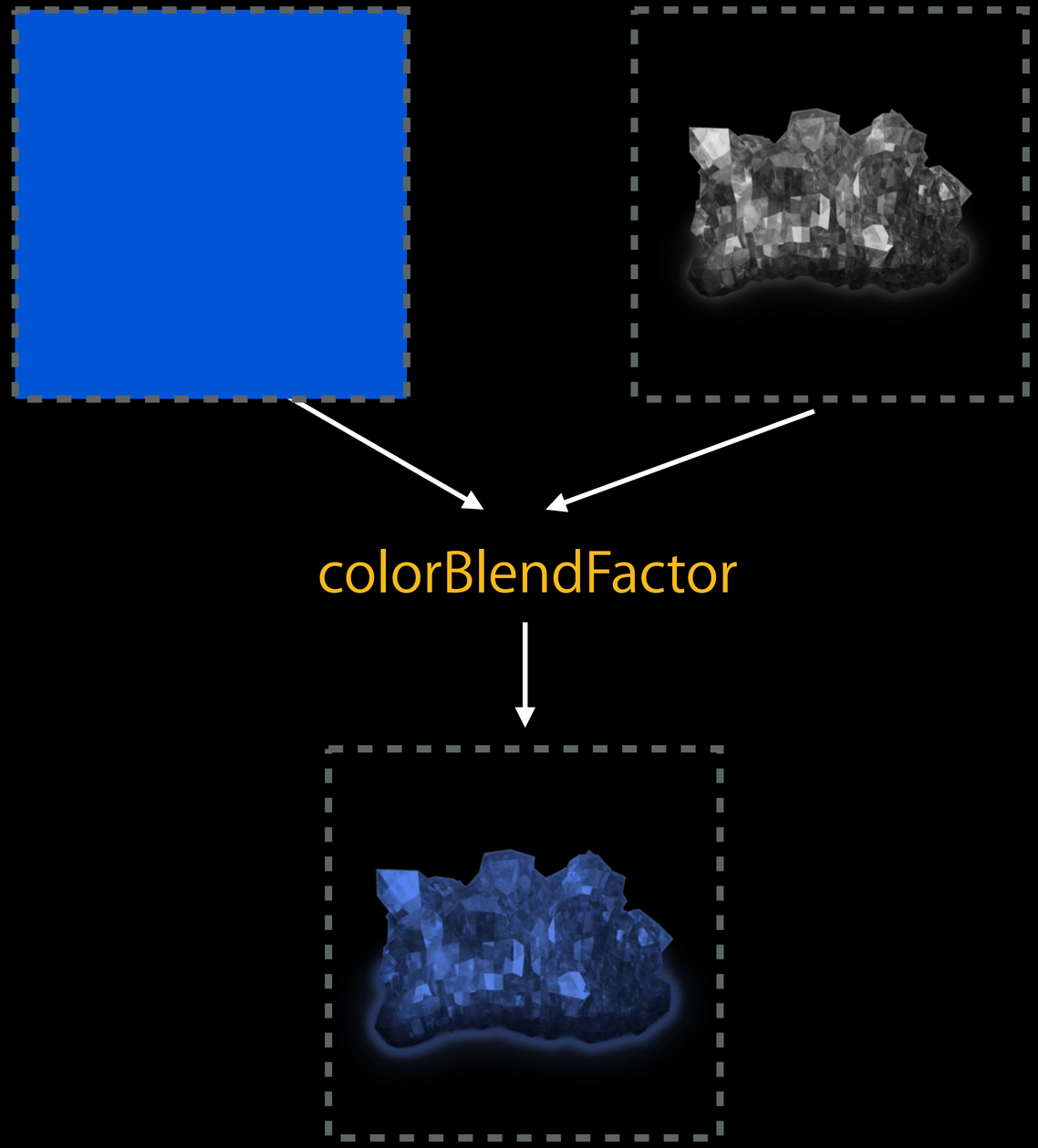
```
/* standard sprite creation from a png file */

SKSpriteNode *sprite = [SKSpriteNode new];
SKTexture *tex = [SKTexture textureWithImageNamed:@"hero.png"];
sprite.texture = tex;
sprite.size = tex.size;
```

```
/* SKSpriteNode convenience method */
SKSpriteNode *sprite = [SKSpriteNode spriteNodeWithImageNamed:@"hero.png"];
```

SKSpriteNode

Sprite Kit MVP

- Combine texture and color
- colorBlendFactor
  - 0.0 is no tinting
  - 1.0 is fully tinted
- Texture color is tinted
- Texture alpha is multiplied
- If texture is nil, color is used



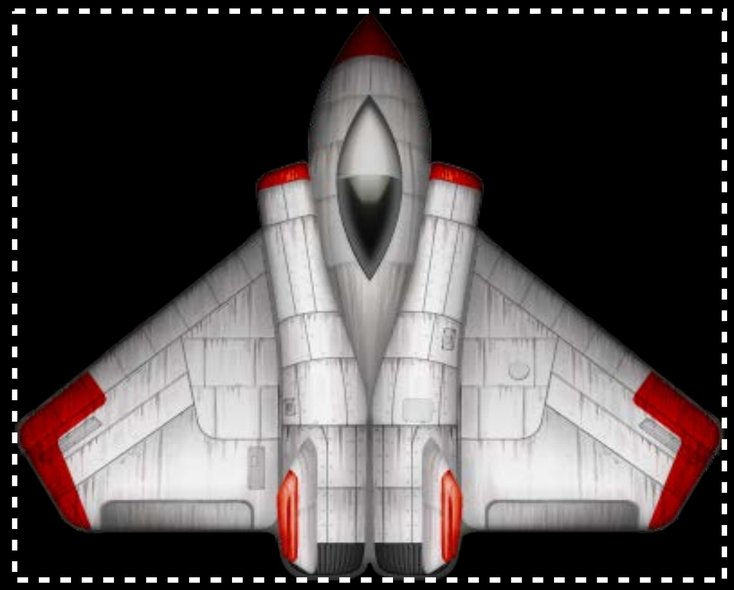
SKSpriteNode

Sprite Kit MVP

```
SKSpriteNode *sprite = [SKSpriteNode spriteNodeWithImageNamed:@"ship.png"];
[self addChild:sprite];

sprite.alpha = 0.5;
sprite.xScale = 0.5;
sprite.zRotation = M_PI / 4.0;
sprite.color = [SKColor colorWithRed:0.0 green:1.0 blue:0.0 alpha:1.0];
sprite.colorBlendFactor = 1.0;
```

SKSpriteNode

Sprite Kit MVP



```
SKSpriteNode *sprite = [SKSpriteNode spriteNodeWithImageNamed:@"ship.png"];
[self addChild:sprite];

sprite.alpha = 0.5;
sprite.xScale = 0.5;
sprite.zRotation = M_PI / 4.0;
sprite.color = [SKColor colorWithRed:0.0 green:1.0 blue:0.0 alpha:1.0];
sprite.colorBlendFactor = 1.0;
```

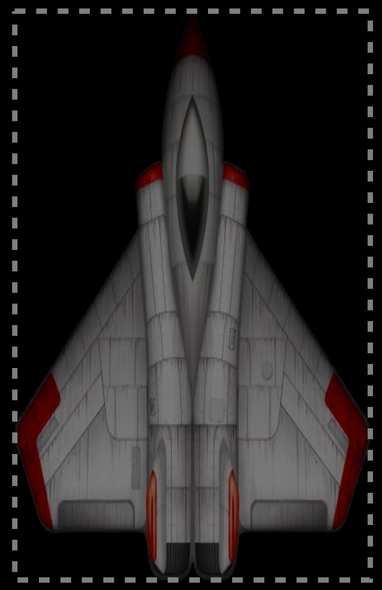
SKSpriteNode

Sprite Kit MVP

```
SKSpriteNode *sprite = [SKSpriteNode spriteNodeWithImageNamed:@"ship.png"];
[self addChild:sprite];

sprite.alpha = 0.5;
sprite.xScale = 0.5;
sprite.zRotation = M_PI / 4.0;
sprite.color = [SKColor colorWithRed:0.0 green:1.0 blue:0.0 alpha:1.0];
sprite.colorBlendFactor = 1.0;
```

SKSpriteNode

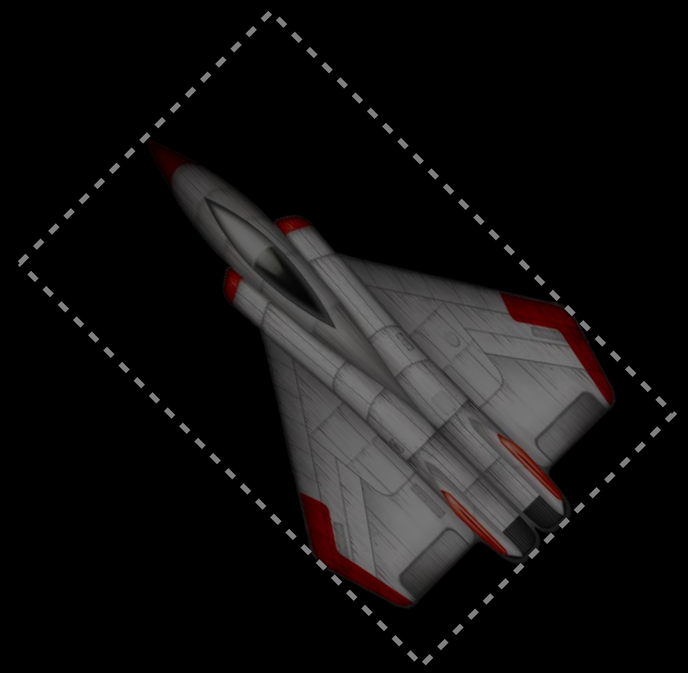
Sprite Kit MVP



```
SKSpriteNode *sprite = [SKSpriteNode spriteNodeWithImageNamed:@"ship.png"];
[self addChild:sprite];

sprite.alpha = 0.5;
sprite.xScale = 0.5;
sprite.zRotation = M_PI / 4.0;
sprite.color = [SKColor colorWithRed:0.0 green:1.0 blue:0.0 alpha:1.0];
sprite.colorBlendFactor = 1.0;
```

SKSpriteNode

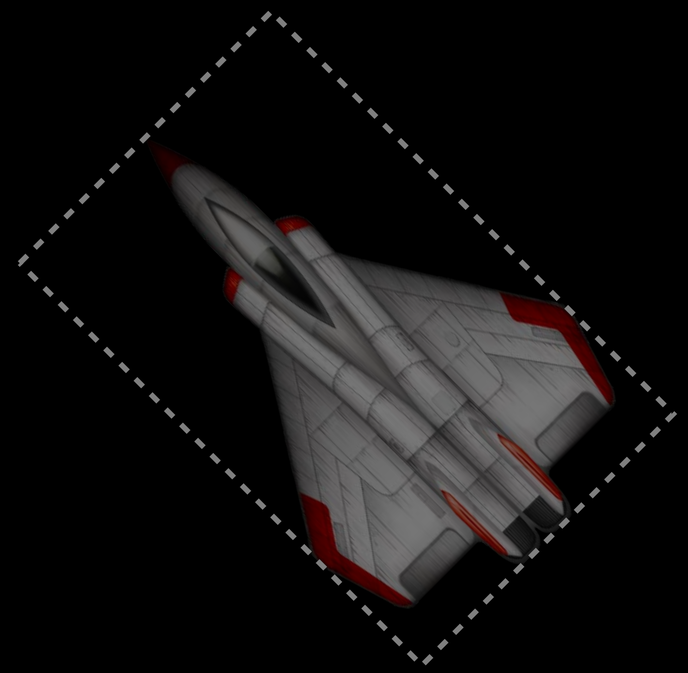
Sprite Kit MVP



```
SKSpriteNode *sprite = [SKSpriteNode spriteNodeWithImageNamed:@"ship.png"];
[self addChild:sprite];

sprite.alpha = 0.5;
sprite.xScale = 0.5;
sprite.zRotation = M_PI / 4.0;
sprite.color = [SKColor colorWithRed:0.0 green:1.0 blue:0.0 alpha:1.0];
sprite.colorBlendFactor = 1.0;
```

SKSpriteNode

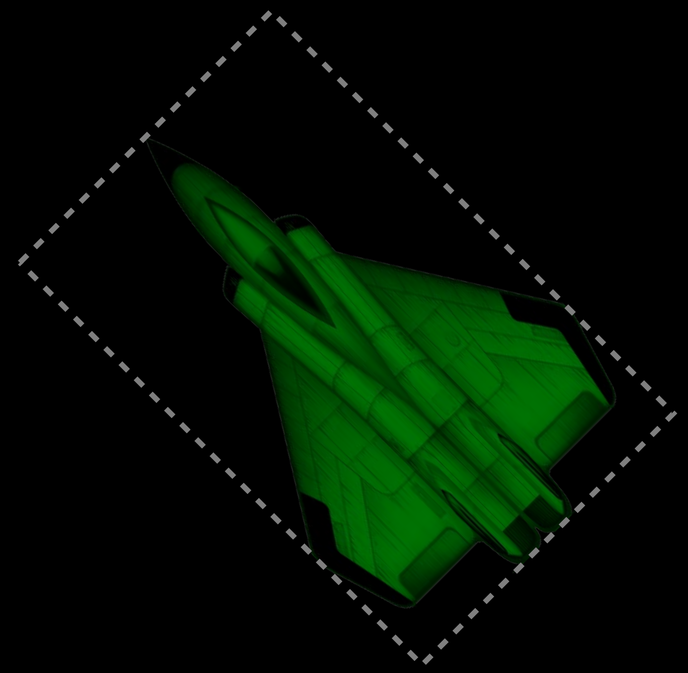
Sprite Kit MVP



```
SKSpriteNode *sprite = [SKSpriteNode spriteNodeWithImageNamed:@"ship.png"];
[self addChild:sprite];

sprite.alpha = 0.5;
sprite.xScale = 0.5;
sprite.zRotation = M_PI / 4.0;
sprite.color = [SKColor colorWithRed:0.0 green:1.0 blue:0.0 alpha:1.0];
sprite.colorBlendFactor = 1.0;
```

SKSpriteNode

Sprite Kit MVP

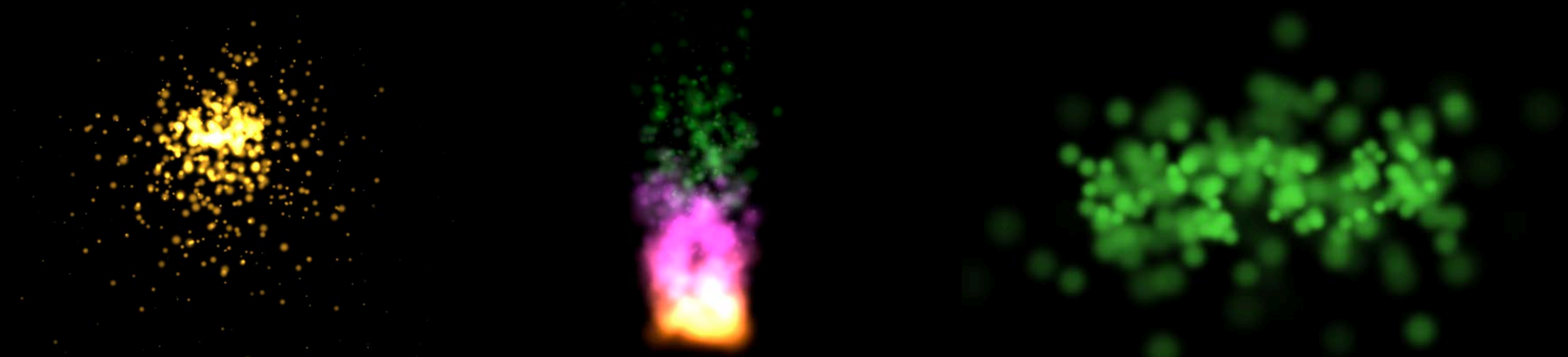


```
SKSpriteNode *sprite = [SKSpriteNode spriteNodeWithImageNamed:@"ship.png"];
[self addChild:sprite];

sprite.alpha = 0.5;
sprite.xScale = 0.5;
sprite.zRotation = M_PI / 4.0;
sprite.color = [SKColor colorWithRed:0.0 green:1.0 blue:0.0 alpha:1.0];
sprite.colorBlendFactor = 1.0;
```

SKEmitterNode

For things that go boom!

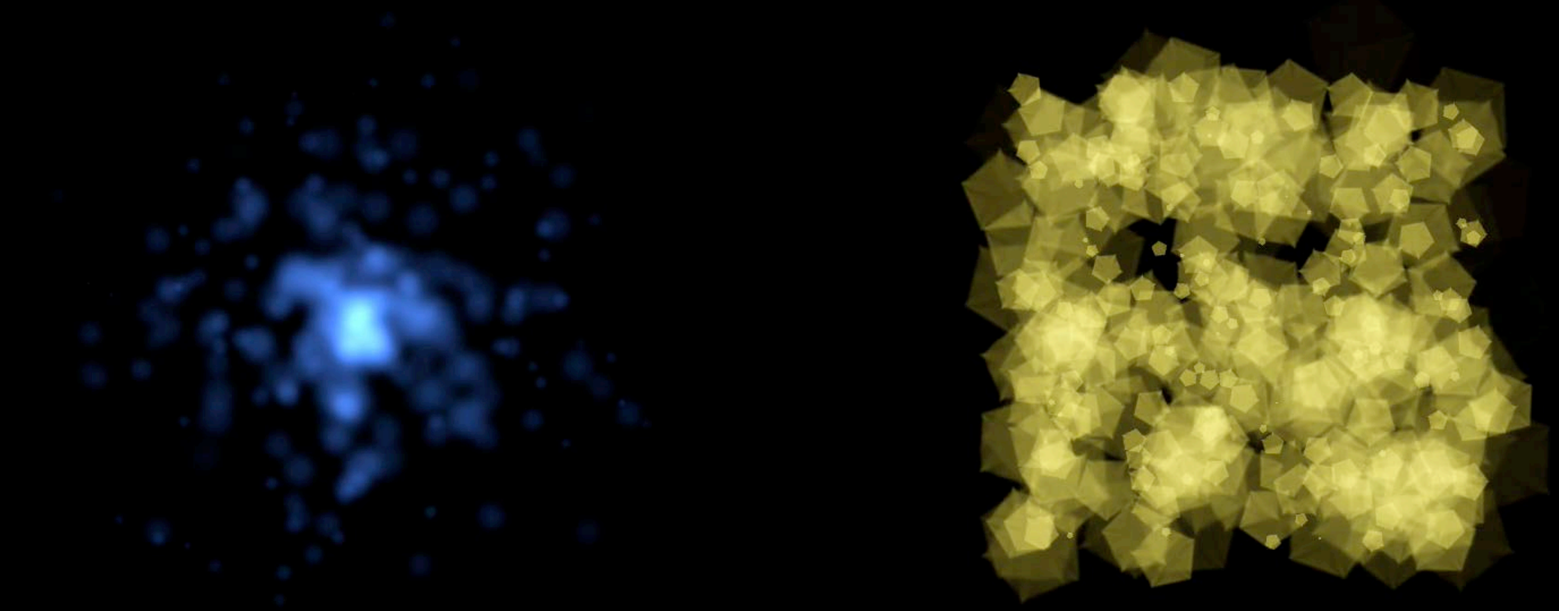
- Full featured 2D particle system
- Standard startValue and speed
- Advanced keyframe sequence controls



SKEmitterNode

For things that go boom!

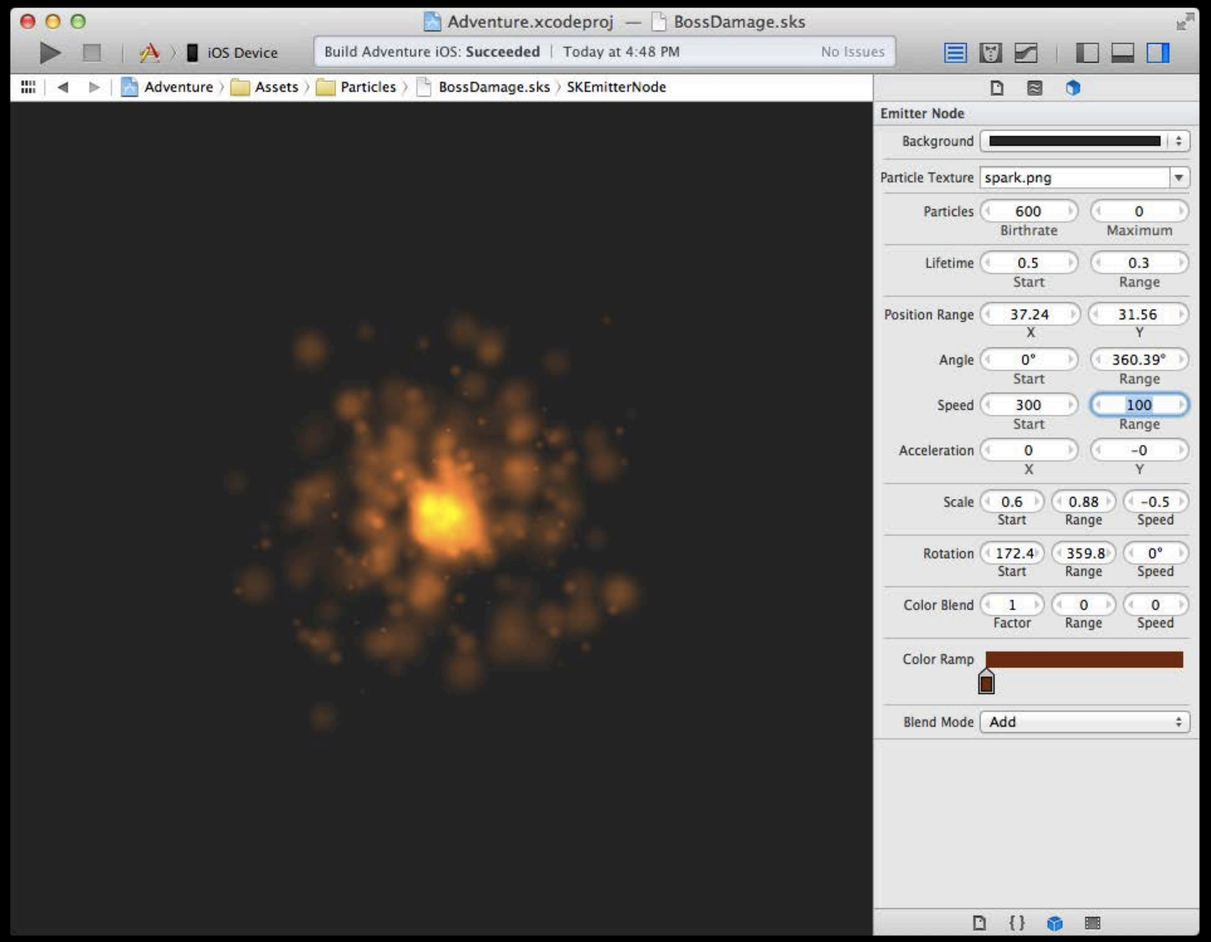
- Texture
- Scale
- Rotation
- Emission angle
- Emission speed
- Blend modes



There are so many things to tweak!

SKEmitterNode

- Data driven particle effects
- Built-in Xcode editor
- Reduce iteration time
- Empower artists



Video in Games

Video as a first class sprite

- Until now video has been:
  - On top your game view
  - Below your game view
  - Roll your own in OpenGL
- In Sprite Kit video is truly a first class sprite

SKVideoNode

Video as a first class sprite

- Easy one-line creation

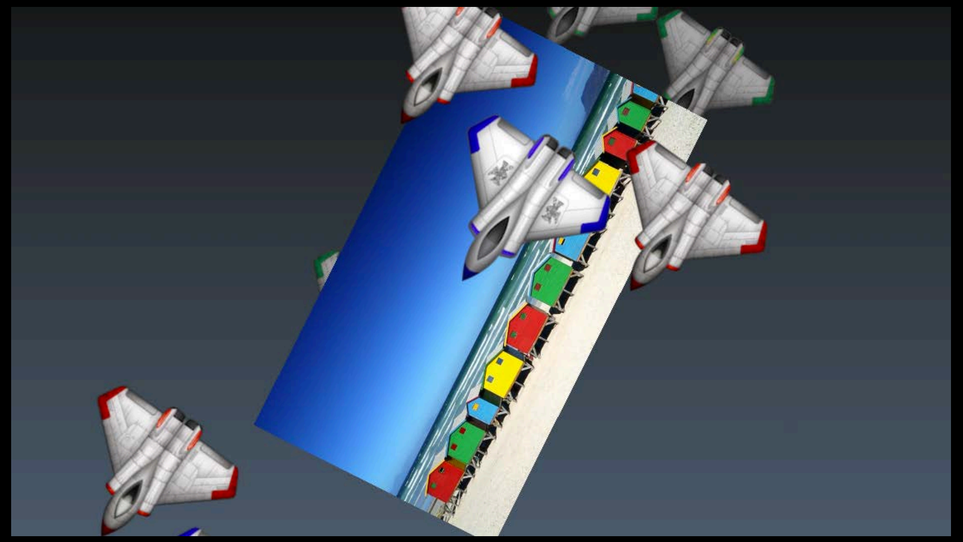
```
[SKVideoNode videoNodeWithVideoFileNamed:@"video.mp4"];
```

- Built on AVPlayer

```
[SKVideoNode videoNodeWithAVPlayer:player];
```

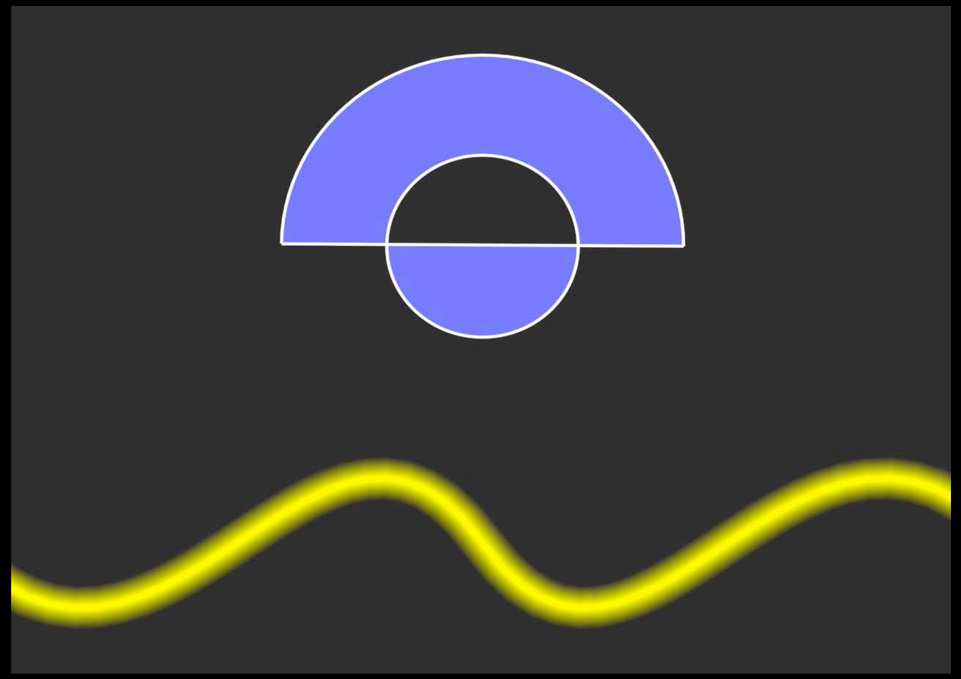
SKVideoNode

- Place anywhere in node tree
- Run SKActions
  - Scale, fade, rotate…
- Video as a level background
- Enable physics on your video



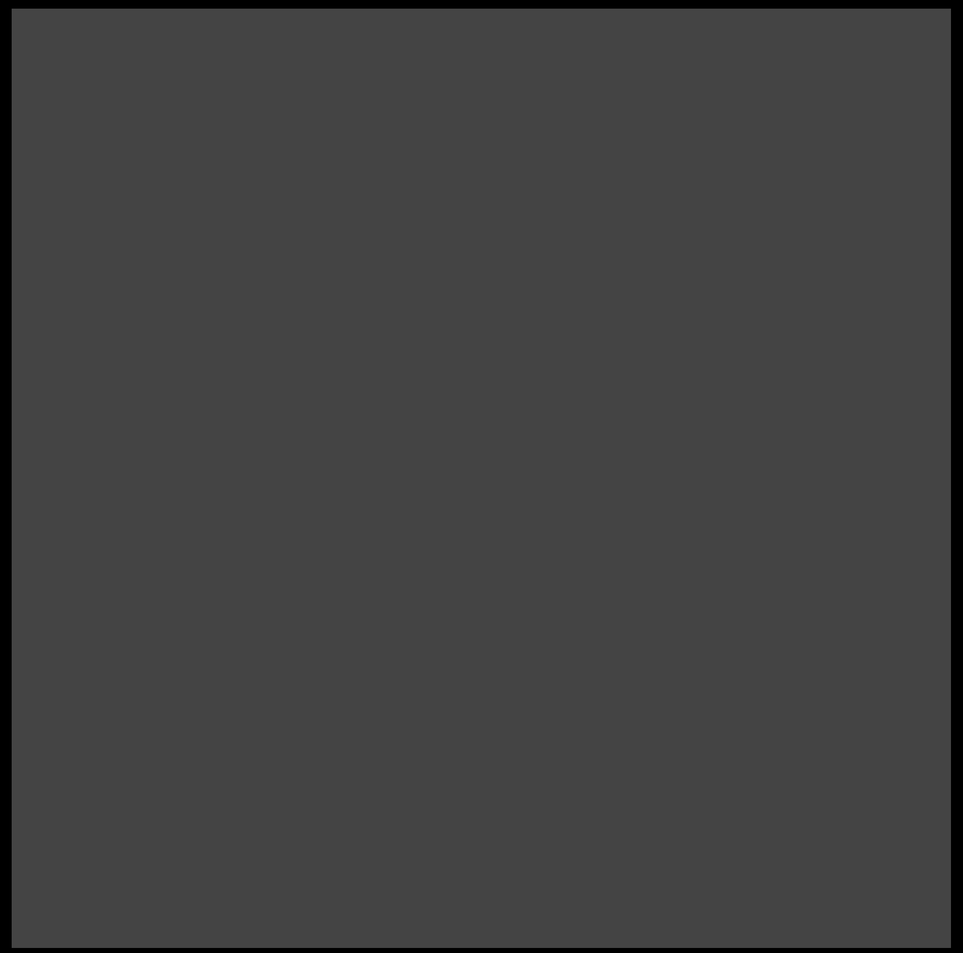
SKShapeNode

- Dynamic shapes
- Any CGPath
- Built for speed
- Rendered in hardware
- Stroke and/or fill
- Add glow effects
- Multiple subpaths



SKLabelNode

- For most text use UIKit/AppKit
- Single line text as a sprite
- Supports all system fonts
- Supports SKActions



SKLabelNode

- For most text use UIKit/AppKit
- Single line text as a sprite
- Supports all system fonts
- Supports SKActions

SKEffectNode

- Flattens children during render
  - `shouldEnableEffects`
  - Group opacity
  - Group blend modes
- Optionally apply a CIFilter
- Can cache via shouldRasterize

SKEffectNode

- Flattens children during render
  - `shouldEnableEffects`
  - Group opacity
  - Group blend modes
- Optionally apply a CIFilter
- Can cache via shouldRasterize

SKCropNode

- Masks content of children
- Mask defined as a SKNode

```
@property (retain) SKNode *maskNode
```

- Transparent area is masked out
- Mask node can have children
- Mask node can run SKActions

SKCropNode

- Masks content of children
- Mask defined as a SKNode

```
@property (retain) SKNode *maskNode
```

- Transparent area is masked out
- Mask node can have children
- Mask node can run SKActions

Actions and Animation

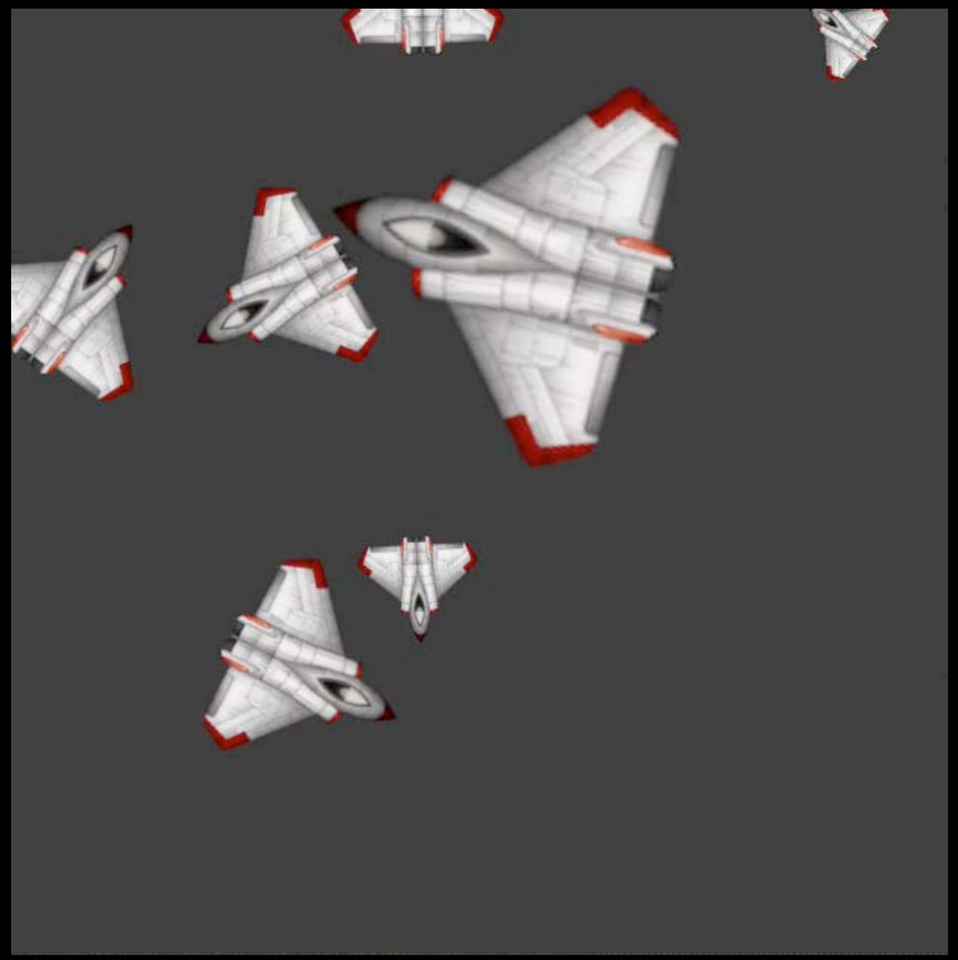
Actions Overview

```
SKAction *a = [SKAction rotateByAngle:M_PI duration:1.0];
```

- We want to design an action system that was super simple to use
  - Single action class—SKAction
  - One line creation
  - Chainable, reusable, readable
  - Actions directly affect the node it is run on
- Almost like a scripting language for Sprite Kit

Basic SKActions

```
[SKAction rotateByAngle:M_PI duration:1.0];

[SKAction moveTo:aCGPoint duration:1.0];

[SKAction fadeAlphaTo:0.75 duration:1.0];

[SKAction scaleBy:2.0 duration:1.0];

[SKAction scaleXBy:1.5 y:0.5 duration:1.0];
```



Running SKActions

Animate your content

- Actions run immediately
- Copy on add
- Removed on completion

```
/* create an SKAction, then add it to your node */
SKAction *rotate = [SKAction rotateByAngle:M_PI duration:1.0];
[sprite runAction:rotate];

/* Or create your action in-line */
[sprite runAction:[SKAction fadeOutWithDuration:1.0]];
```

Repeating Actions

- Repeating an action:

```
SKAction *spin = [SKAction rotateByAngle:2 * M_PI duration:1.0];
SKAction *spinThreeTimes = [SKAction repeatAction: rotate count:3];
```

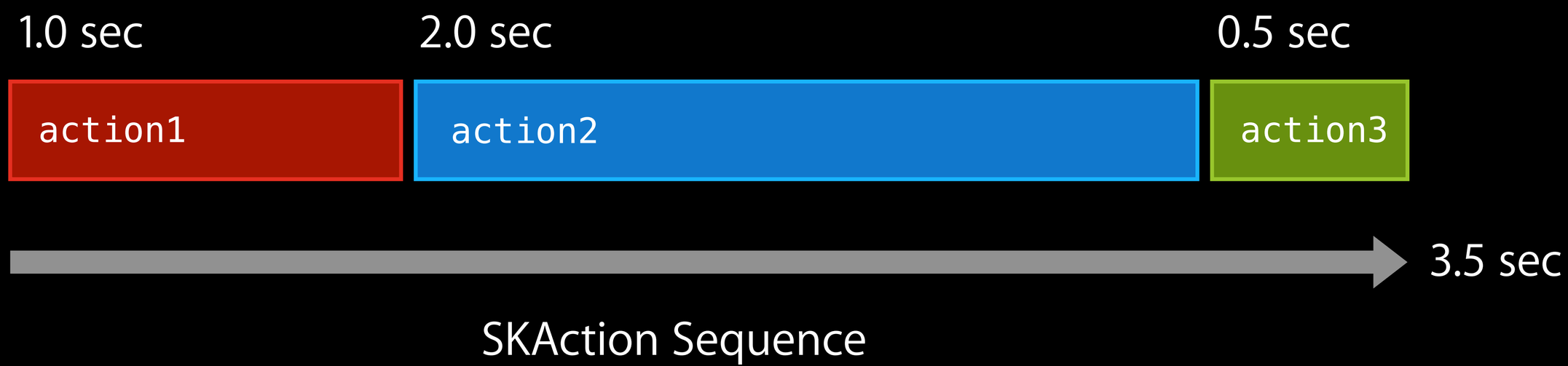
- Repeating an action forever:

```
SKAction *spinForever = [SKAction repeatActionForever:rotate];
```

Sequences

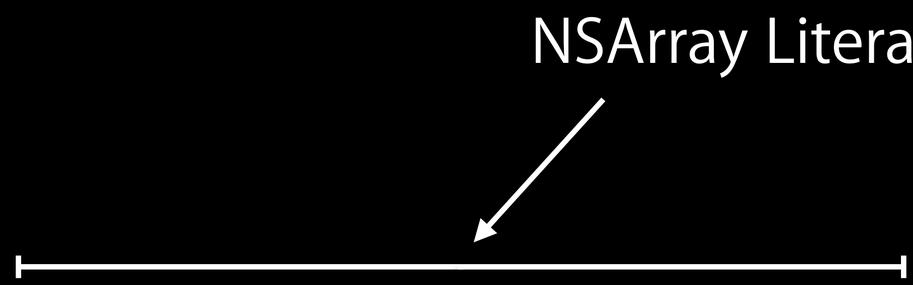
Reuse the basic building blocks

```
[myNode runAction: [SKAction sequence:@[action1, action2, action3]] ];
```

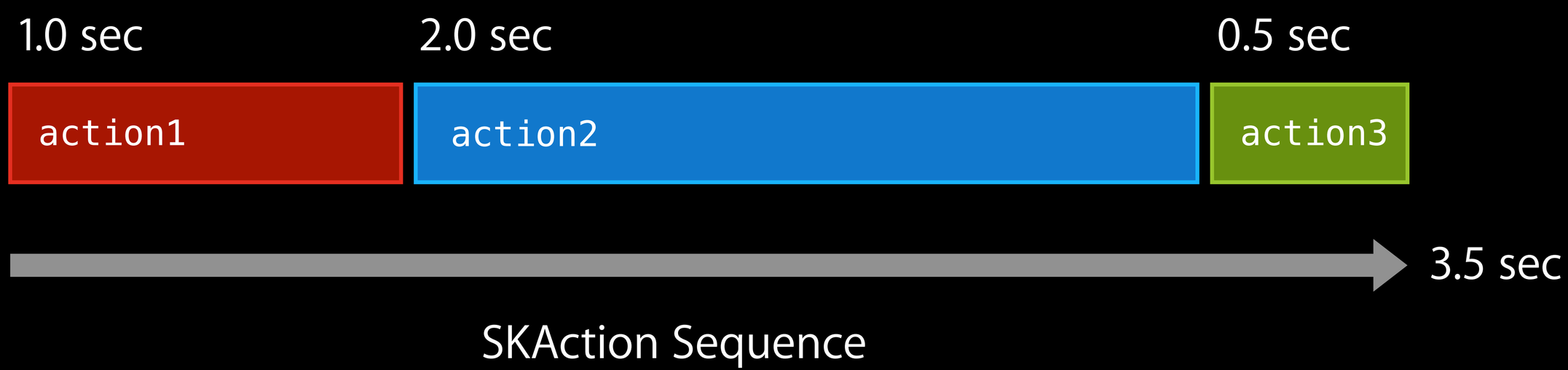


Sequences

Reuse the basic building blocks



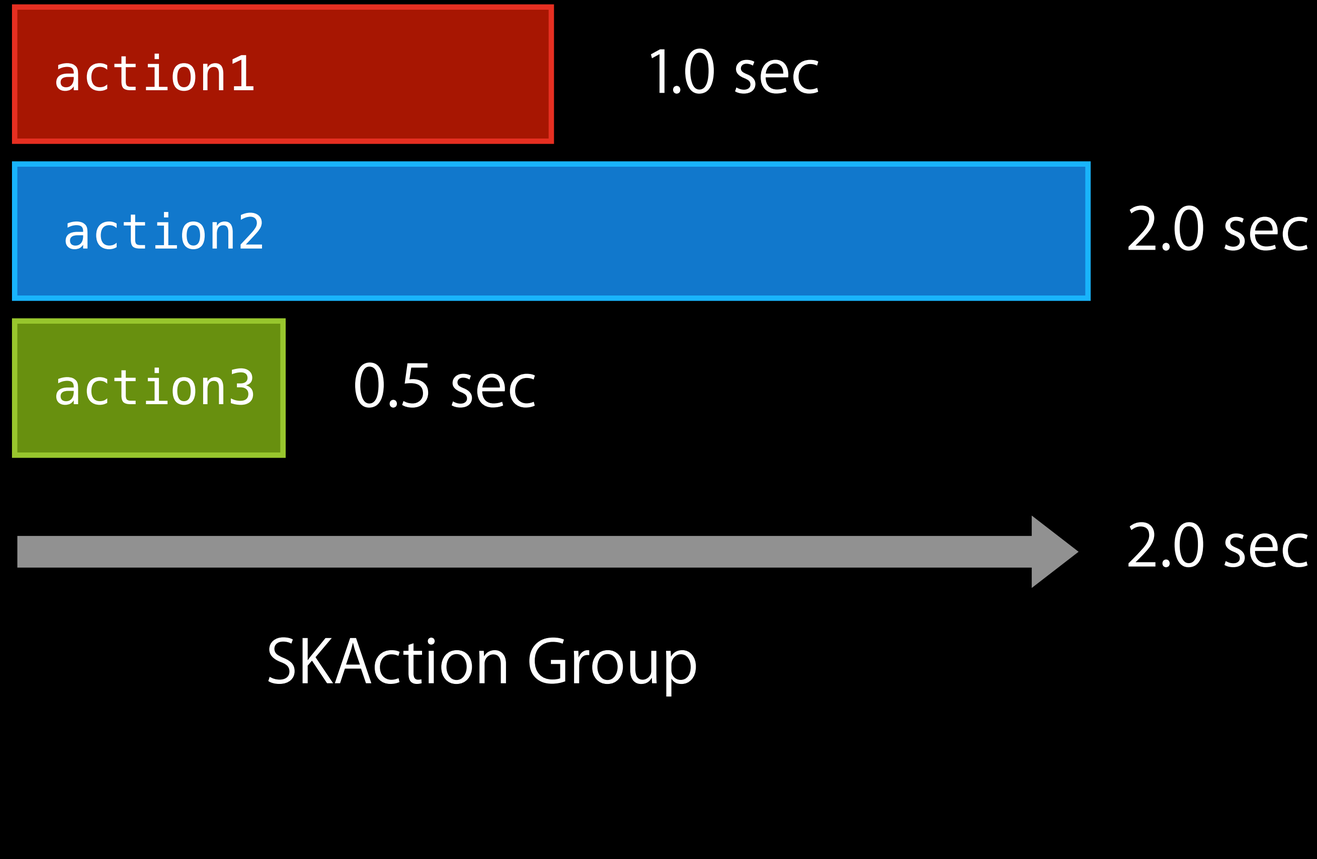

```
[myNode runAction: [SKAction sequence:@[action1, action2, action3]] ];
```



Groups

Reuse the basic building blocks

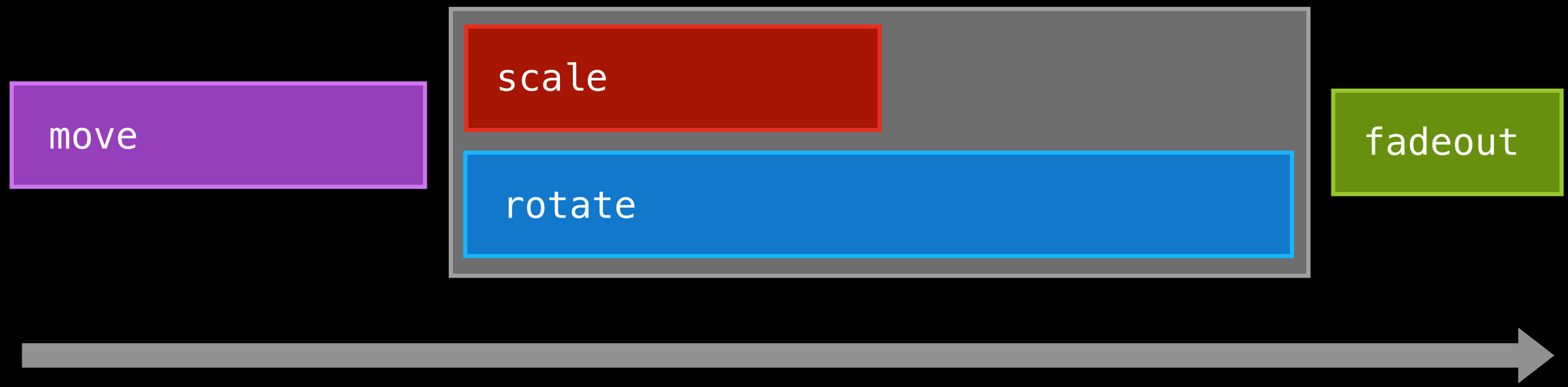
```
[myNode runAction: [SKAction group:@[action1, action2, action3]] ];
```



Compound Actions

Sequences of groups

```
SKAction *group = [SKAction group:@[scale, rotate]];

[myNode runAction: [SKAction sequence:@[move, group, fadeout]] ];
```



Sequence with a Group

Timing of Actions

Keep it simple

- No explicit timing
- Utilize sequences

```
SKAction *wait = [SKAction waitForDuration:1.0];
[myNode runAction: [SKAction sequence:@[wait, action1]] ];
```

Sequence with Wait

Specialty Actions

Specialty SKActions

Animate

```
[SKAction animateWithTextures:@[tex0, tex1, tex2] timePerFrame:0.1];
```

Specialty SKActions

Animate

```
[SKAction animateWithTextures:@[tex0, tex1, tex2] timePerFrame:0.1];
```

Specialty SKActions

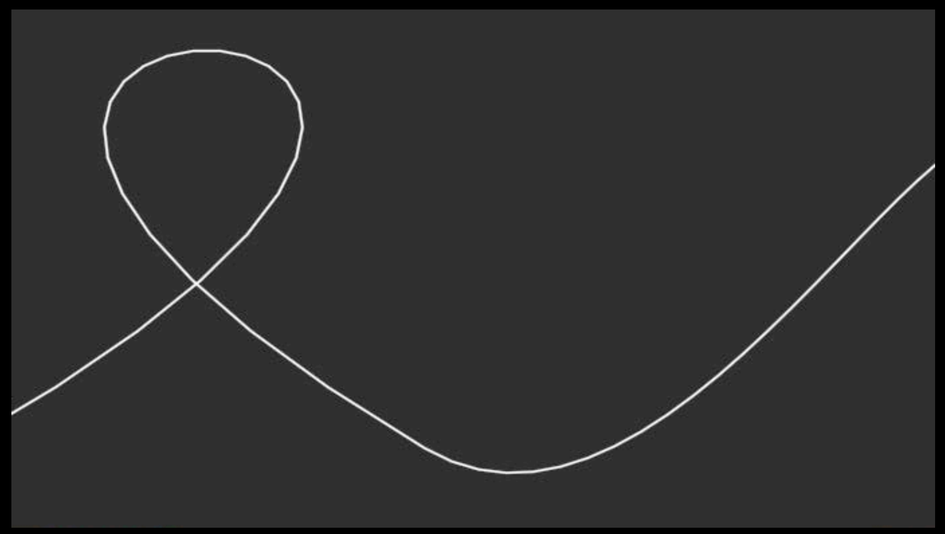
Follow path

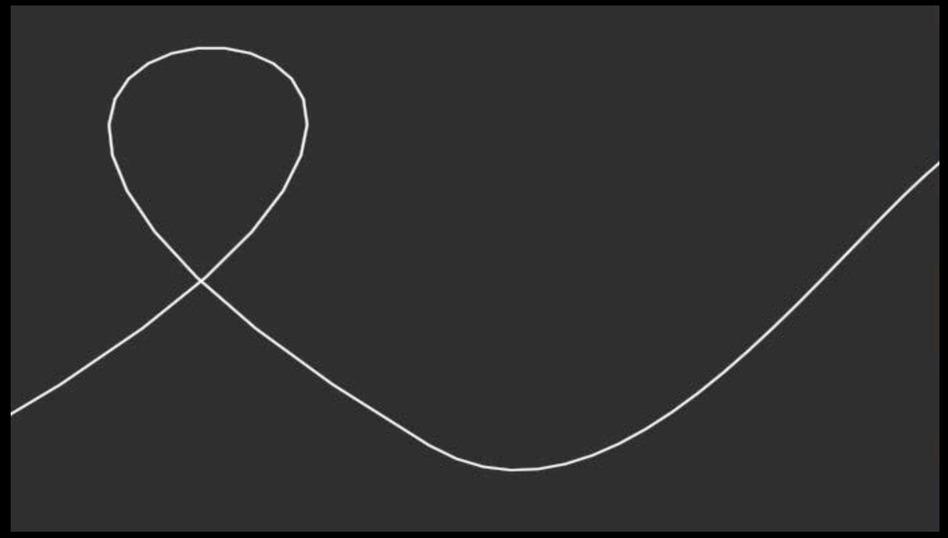
```
[SKAction followPath:myPath duration:2.5]

[SKAction followPath:myPath asOffset:YES orientToPath:NO duration:5.0];
```

Specialty SKActions

Follow path

```
[SKAction followPath:myPath duration:2.5]

[SKAction followPath:myPath asOffset:YES orientToPath:NO duration:5.0];
```

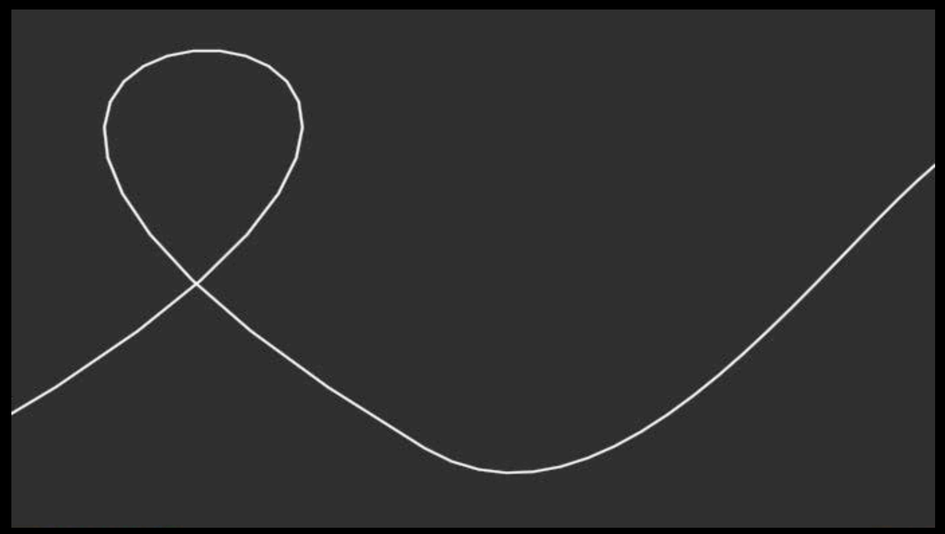


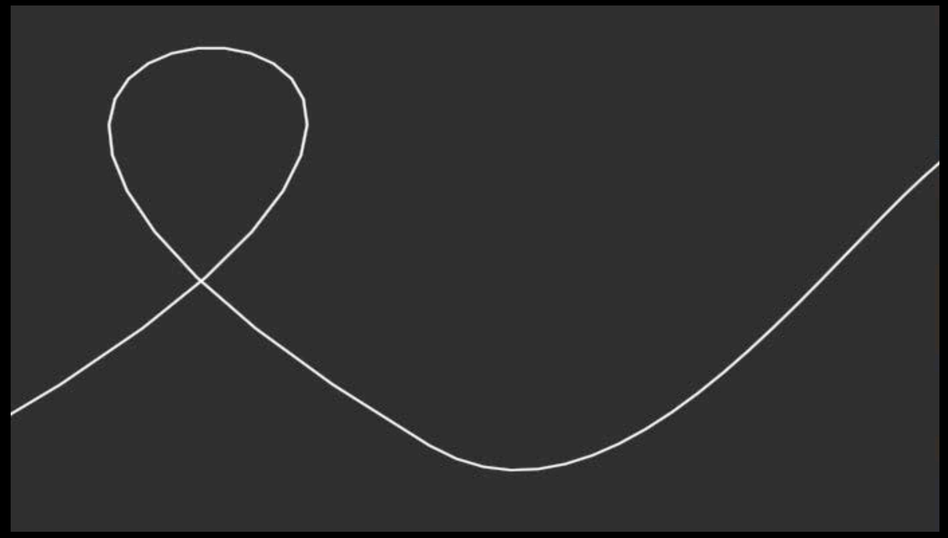


Specialty SKActions

Follow path

```
[SKAction followPath:myPath duration:2.5]

[SKAction followPath:myPath asOffset:YES orientToPath:NO duration:5.0];
```





Specialty SKActions

Remove from parent

```
/* zero duration */

[SKAction removeFromParent];
```

```
/* fade out a goblin and then remove it */

SKAction *fade = [SKAction fadeOutWithDuration:1.0];
SKAction *scale = [SKAction scaleBy:0.5 duration:1.0];

SKAction *group = [SKAction group:@[fade, scale]];
SKAction *remove = [SKAction removeFromParent];


[goblin runAction:[SKAction sequence:@[group, remove]];
```

Specialty SKActions

Sound effects

- Great for short sound effects
- Sound always plays to completion
- Use AVFoundation for longer sounds or playback control

```
/* zero duration, starts playback */
[SKAction playSoundFileNamed:@”pew_pew.caf” waitForCompletion:NO]

/* starts playback then waits for the duration of the sound */
[SKAction playSoundFileNamed:@”kaboom.caf” waitForCompletion:YES]
```

Specialty SKActions

Run block

```
/* zero duration, fires once */

[SKAction runBlock:^{ doSomething(); }]
```

```
/* show game over menu after character death animation */

SKAction *fadeOut = [SKAction fadeOutWithDuration:1.0];
SKAction *remove = [SKAction removeFromParent];
SKAction *showMenu = [SKAction runBlock:^{ [self showGameOverMenu]; }];

[heroSprite runAction: [SKAction sequence:@[fadeOut, showMenu, remove]] ];
```

Specialty SKActions

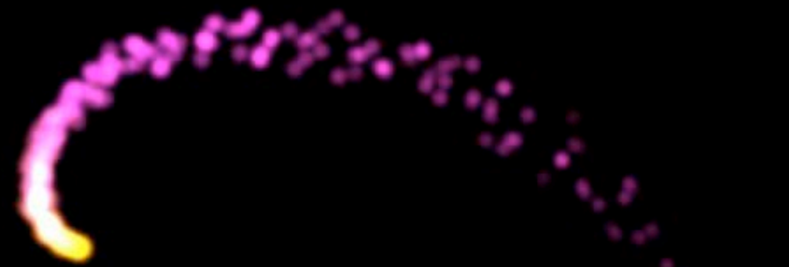
Custom actions

```
[SKAction customActionWithDuration:dur actionBlock:^(SKNode *n, CGFloat t) {
     CGFloat ratio = t / dur;
     SKEmitterNode *en = (SKEmitterNode*)n;
     en.emissionAngle = ratio * 2 * M_PI;
}];
```

Specialty SKActions

Custom actions

```
[SKAction customActionWithDuration:dur actionBlock:^(SKNode *n, CGFloat t) {
     CGFloat ratio = t / dur;
     SKEmitterNode *en = (SKEmitterNode*)n;
     en.emissionAngle = ratio * 2 * M_PI;
}];
```



Actions

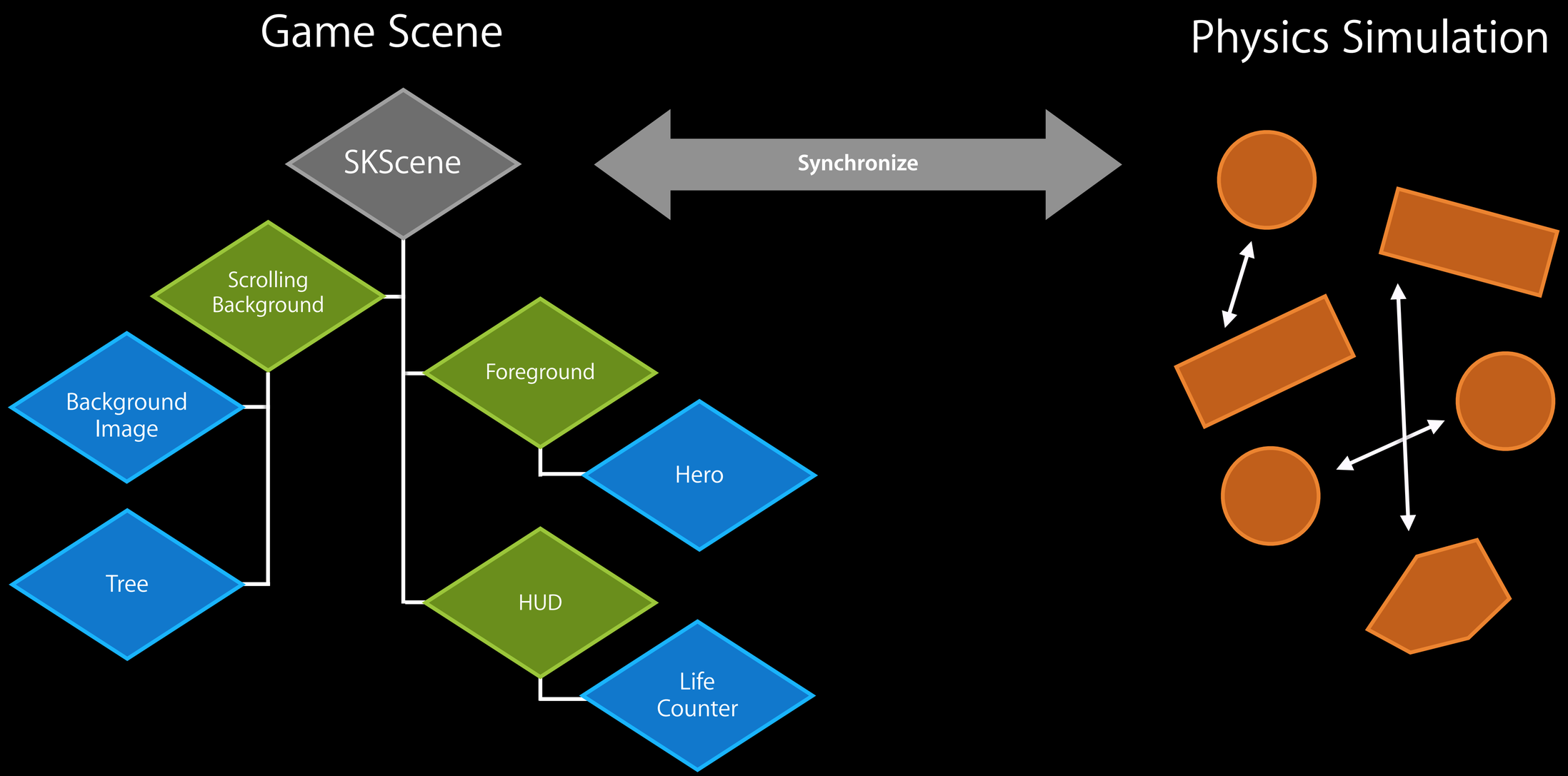
Actions

```
moveByX:(CGFloat)deltaX y:(CGFloat)deltaY duration:(NSTimeInterval)sec;
moveTo:(CGPoint)location duration:(NSTimeInterval)sec;
moveToX:(CGFloat)x duration:(NSTimeInterval)sec;
moveToY:(CGFloat)y duration:(NSTimeInterval)sec;
rotateByAngle:(CGFloat)radians duration:(NSTimeInterval)sec;
rotateToAngle:(CGFloat)radians duration:(NSTimeInterval)sec;
resizeByWidth:(CGFloat)width height:(CGFloat)height duration:(NSTimeInterval)duration;
resizeToWidth:(CGFloat)width height:(CGFloat)height duration:(NSTimeInterval)duration;
resizeToWidth:(CGFloat)width duration:(NSTimeInterval)duration;
resizeToHeight:(CGFloat)height duration:(NSTimeInterval)duration;
scaleBy:(CGFloat)scale duration:(NSTimeInterval)sec;
scaleXBy:(CGFloat)xScale y:(CGFloat)yScale duration:(NSTimeInterval)sec;
scaleTo:(CGFloat)scale duration:(NSTimeInterval)sec;
scaleXTo:(CGFloat)xScale y:(CGFloat)yScale duration:(NSTimeInterval)sec;
scaleXTo:(CGFloat)scale duration:(NSTimeInterval)sec;
scaleYTo:(CGFloat)scale duration:(NSTimeInterval)sec;
sequence:(NSArray *)actions;
group:(NSArray *)actions;
repeatAction:(SKAction *)action count:(NSUInteger)count;
repeatActionForever:(SKAction *)action;
fadeInWithDuration:(NSTimeInterval)sec;
fadeOutWithDuration:(NSTimeInterval)sec;
fadeAlphaBy:(CGFloat)factor duration:(NSTimeInterval)sec;
fadeAlphaTo:(CGFloat)alpha duration:(NSTimeInterval)sec;
setTexture:(SKTexture *)texture;
animateWithTextures:(NSArray *)textures timePerFrame:(NSTimeInterval)sec;
animateWithTextures:(NSArray *)textures timePerFrame:(NSTimeInterval)sec resize:...;
playSoundFileNamed:(NSString*)soundFile waitForCompletion:(BOOL)wait;
```

Built in Physics Simulation

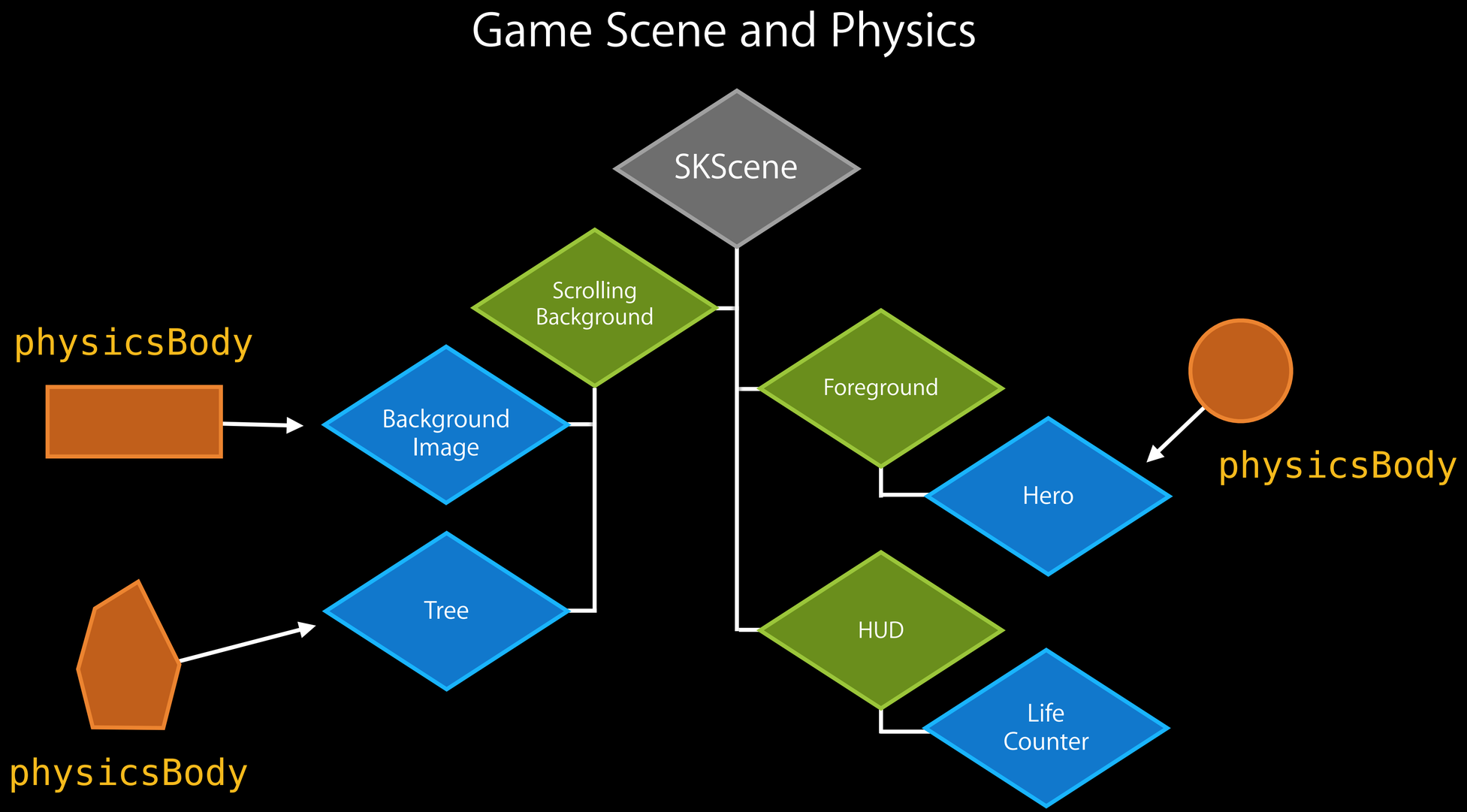
Physics in Sprite Kit

Truly integrated physics



Physics in Sprite Kit

Truly integrated physics



Physics in Sprite Kit

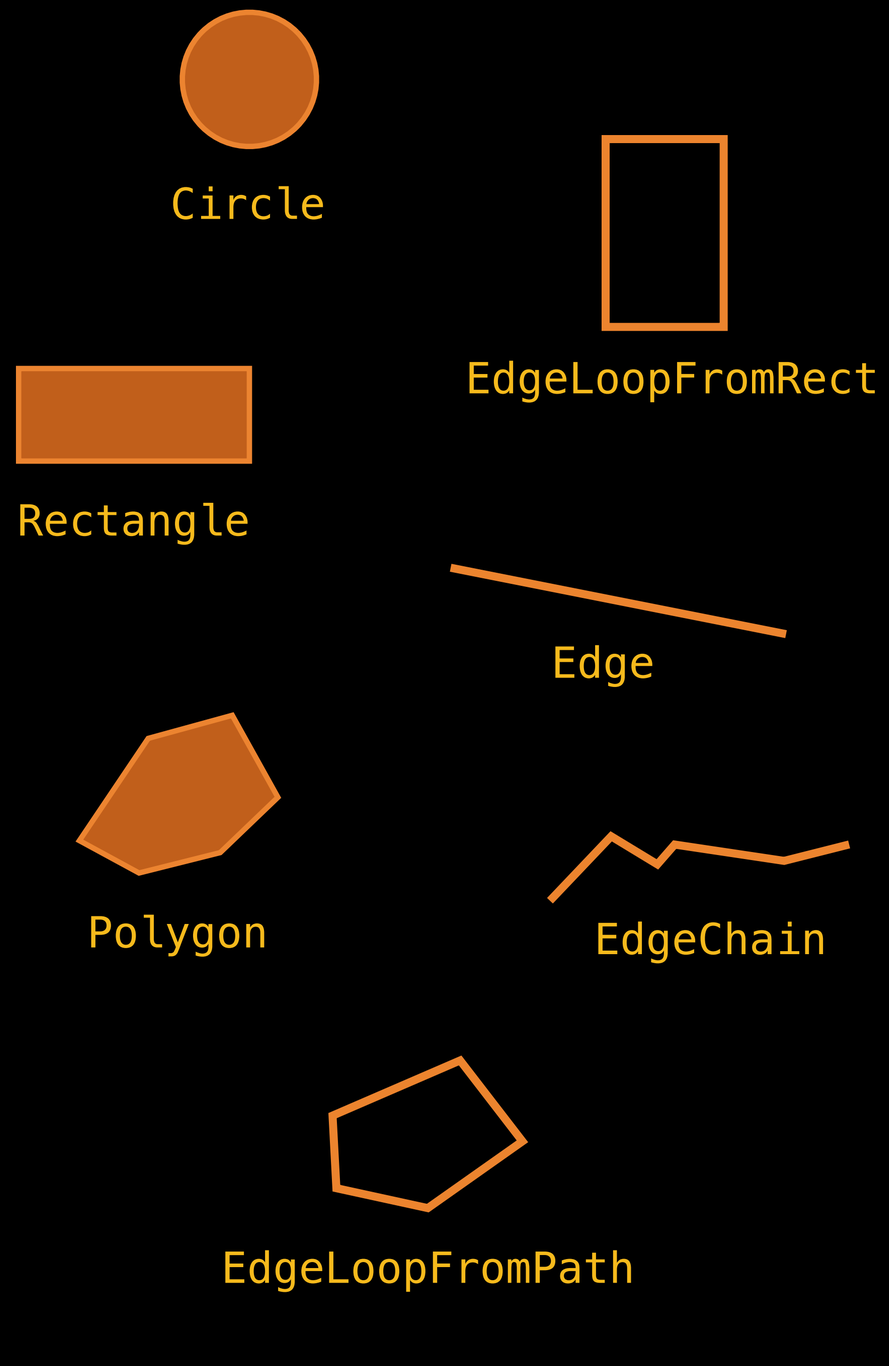
Truly integrated physics

- Built right into Sprite Kit
- We do the synchronization
- Not a global on/off switch
- Enabled on a node-by-node basis
- No performance penalty for what you're not using

SKPhysicsBody

Add physics to your nodes

```
/* solid circle centered at node's anchorPoint */
[SKPhysicsBody bodyWithCircleOfRadius:50];

/* hollow rect relative to node's anchorPoint */
[SKPhysicsBody bodyWithEdgeLoopFromRect:rect];

/* solid rect centered at node's anchorPoint */
[SKPhysicsBody bodyWithRectangleOfSize:size];

/* zero-width edge relative to node's anchorPoint */
[SKPhysicsBody bodyWithEdgeFromPoint:p0 toPoint:p1];

/* solid polygon relative to node's anchorPoint */
[SKPhysicsBody bodyWithPolygonFromPath:path];

/* zero-width edge relative to node's anchorPoint */
[SKPhysicsBody bodyWithEdgeChainFromPath:path];

/* hollow polygon relative to node's anchorPoint */
[SKPhysicsBody bodyWithEdgeLoopFromPath:path];
```



SKPhysicsBody

Add physics to your nodes

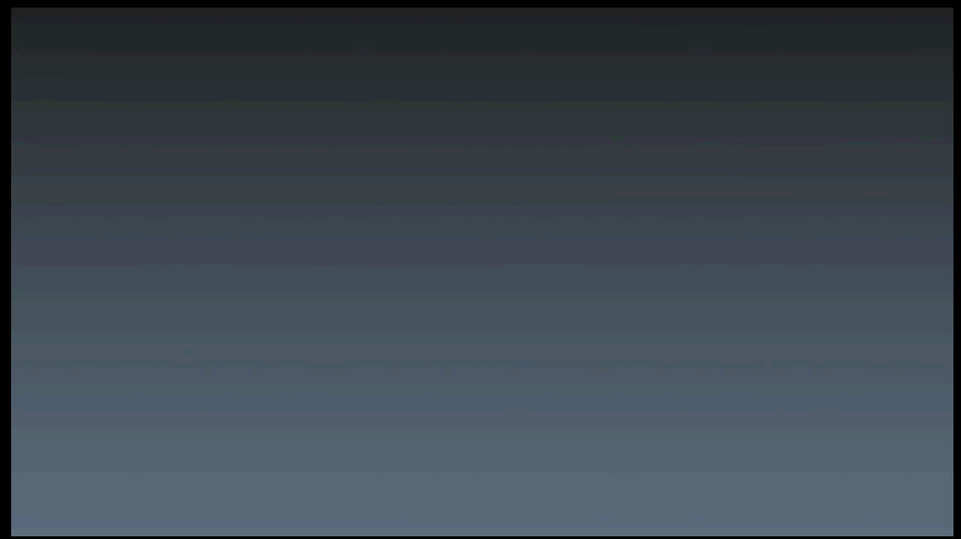
```
/* create a circular a physicsBody */
[SKPhysicsBody bodyWithCircleOfRadius:50];

/* to enable physics, set a physicsBody */
mySprite.physicsBody = myPhysicsBody;
```

```
/* add a sprite to the scene and enable physics on it */
SKSpriteNode *sprite = [SKSpriteNode spriteNodeWithImageNamed:@"ball.png"];
sprite.physicsBody = [SKPhysicsBody bodyWithCircleOfRadius:sprite.size.width * 0.5];

[self addChild:sprite];
```

SKPhysicsBody

Add physics to your nodes

```
/* create a circular a physicsBody */
[SKPhysicsBody bodyWithCircleOfRadius:50];

/* to enable physics, set a physicsBody */
mySprite.physicsBody = myPhysicsBody;
```



```
/* add a sprite to the scene and enable physics on it */
SKSpriteNode *sprite = [SKSpriteNode spriteNodeWithImageNamed:@"ball.png"];
sprite.physicsBody = [SKPhysicsBody bodyWithCircleOfRadius:sprite.size.width * 0.5];

[self addChild:sprite];
```

SKPhysicsBody

Add physics to your nodes

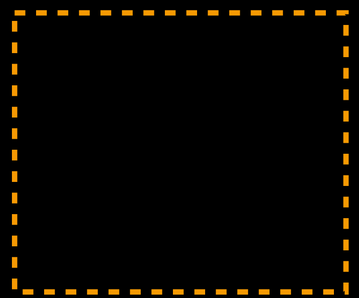
- Use an edgeLoop for a hollow rect

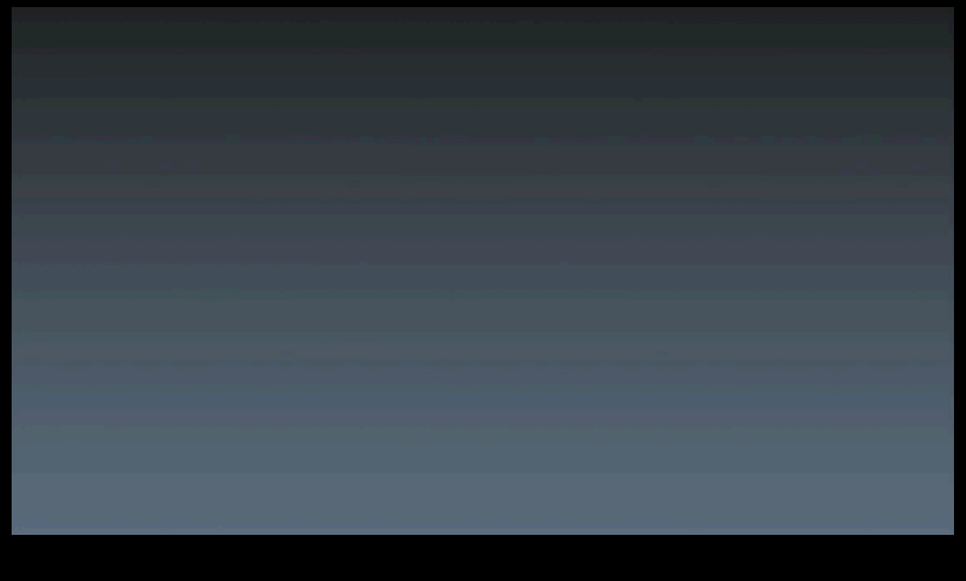
```
/* set the scene’s physicsBody to be an edge loop */
self.physicsBody = [SKPhysicsBody bodyWithEdgeLoopFromRect:self.frame];

SKSpriteNode *sprite = [SKSpriteNode spriteNodeWithImageNamed:@"sphere.png"];
sprite.physicsBody = [SKPhysicsBody bodyWithCircleOfRadius:sprite.size.width * 0.5];

[self addChild:sprite];
```

SKPhysicsBody

Add physics to your nodes

- Use an edgeLoop for a hollow rect





```
/* set the scene’s physicsBody to be an edge loop */
self.physicsBody = [SKPhysicsBody bodyWithEdgeLoopFromRect:self.frame];

SKSpriteNode *sprite = [SKSpriteNode spriteNodeWithImageNamed:@"sphere.png"];
sprite.physicsBody = [SKPhysicsBody bodyWithCircleOfRadius:sprite.size.width * 0.5];

[self addChild:sprite];
```

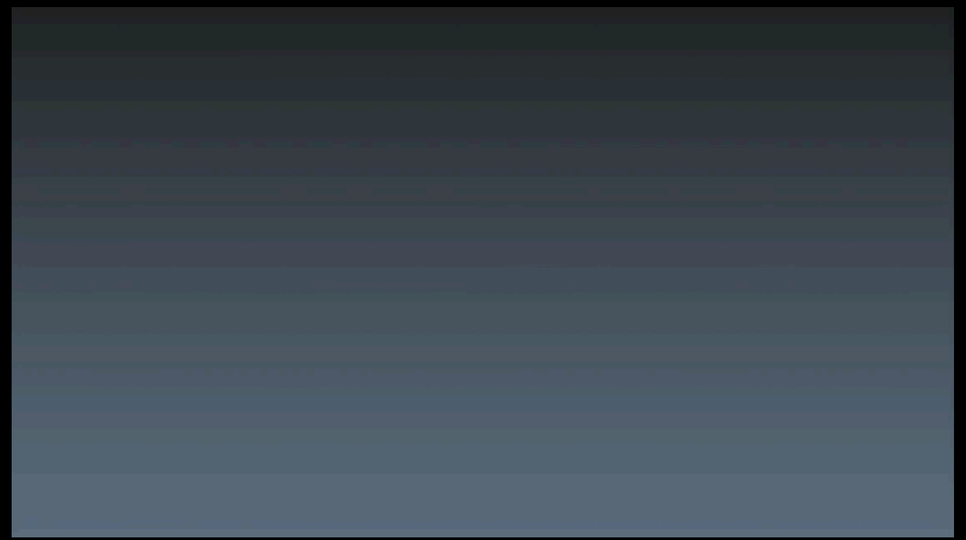
SKPhysicsWorld

- Each SKScene has a physicsWorld
- Perform hit tests, ray casts
- Add joints
- Change gravity

```
/* default gravity, things fall down */
self.physicsWorld.gravity = CGPointMake(0.0, -9.8);

/* make things fall up! */
self.physicsWorld.gravity = CGPointMake(0.0, +9.8);
```

SKPhysicsWorld

- Each SKScene has a physicsWorld
- Perform hit tests, ray casts
- Add joints
- Change gravity



```
/* default gravity, things fall down */
self.physicsWorld.gravity = CGPointMake(0.0, -9.8);

/* make things fall up! */
self.physicsWorld.gravity = CGPointMake(0.0, +9.8);
```

PhysicsWorld Contact Delegate

```
self.physicsWorld.contactDelegate = myContactDelegate;
```

```
@protocol SKPhysicsContactDelegate<NSObject>
  - (void)didBeginContact:(SKPhysicsContact *)contact;
  - (void)didEndContact:(SKPhysicsContact *)contact;
@end
```

SKPhysicsContact

```
@interface SKPhysicsContact : NSObject

  /* the two physics bodies that contacted */
  @property (readonly) SKPhysicsBody *bodyA;
  @property (readonly) SKPhysicsBody *bodyB;

  /* point of first contact */
  @property (readonly) CGPoint contactPoint;

  /* magnitude of collision impulse at that point */
  @property (readonly) CGFloat collisionImpulse;

@end
```

Collisions, Raycasts, and More

```
myScene.physicsWorld.contactDelegate = self;


- (void)didBeginContact:(SKPhysicsContact *)contact {
    if (contact.bodyA.node == heroSprite || contact.bodyB.node == heroSprite)
    {
        // Hero hit something!
    }
}
```

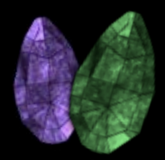
Collision Groups

Define logical groups

Player1

Player2

Baddies



Collectable Power Ups

Collision Groups

Define logical groups

```
/**
 Defines what logical 'categories' this body belongs too. Defaults to all
bits set (all categories).
 */
@property (assign) uint32_t categoryBitMask;

/**
 Defines what logical 'categories' of bodies this body responds to collisions
with. Defaults to all bits set (all categories).
 */
@property (assign) uint32_t collisionBitMask;

/**
 Defines what logical 'categories' of bodies this body generates intersection
notifications with. Defaults to all bits cleared (no categories).
 */
@property (assign) uint32_t contactTestBitMask;
```

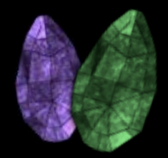
Collision Groups

Define logical groups

Baddies

Player1

Player2



Collectable Power Ups

Collision Groups

Define logical groups

```
const uint32_t GOOD_GUYS   = 0x1 << 0;
const uint32_t BAD_GUYS    = 0x1 << 1;
const uint32_t POWER_UPS   = 0x1 << 2;
```

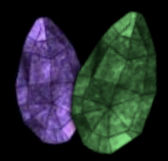
Baddies

BAD_GUYS

Player1

Player2

GOOD_GUYS



Collectable Power Ups

POWER_UPS

Collision Groups

Define logical groups

```
const uint32_t GOOD_GUYS   = 0x1 << 0;
const uint32_t BAD_GUYS    = 0x1 << 1;
const uint32_t POWER_UPS   = 0x1 << 2;
```

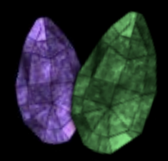
Baddies

BAD_GUYS

Player1

Player2

GOOD_GUYS



Collectable Power Ups

POWER_UPS

Collision Groups

Define logical groups

```
const uint32_t GOOD_GUYS    = 0x1 << 0;
const uint32_t BAD_GUYS     = 0x1 << 1;
const uint32_t POWER_UPS    = 0x1 << 2;
```

Baddies

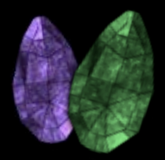
BAD_GUYS

Collide

Player1

Player2

GOOD_GUYS



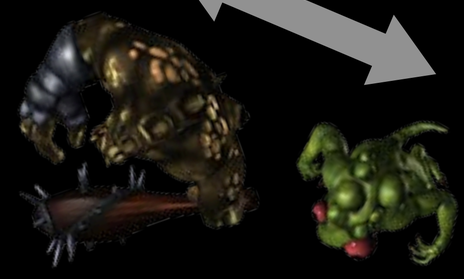
Collectable Power Ups

POWER_UPS

Collision Groups

Define logical groups

```
const uint32_t GOOD_GUYS    = 0x1 << 0;
const uint32_t BAD_GUYS     = 0x1 << 1;
const uint32_t POWER_UPS    = 0x1 << 2;
```



Baddies

BAD_GUYS

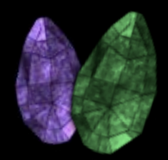
Collide

Player1

Player2

GOOD_GUYS

Collide



Collectable Power Ups

POWER_UPS

Collision Groups

Define logical groups

```
const uint32_t GOOD_GUYS   = 0x1 << 0;
const uint32_t BAD_GUYS    = 0x1 << 1;
const uint32_t POWER_UPS   = 0x1 << 2;
```

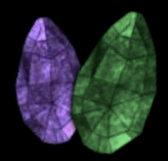
Baddies

BAD_GUYS

Player1

Player2

GOOD_GUYS



Collectable Power Ups

POWER_UPS

Collision Groups

Define logical groups

```
const uint32_t GOOD_GUYS    = 0x1 << 0;
const uint32_t BAD_GUYS     = 0x1 << 1;
const uint32_t POWER_UPS    = 0x1 << 2;
```

Baddies

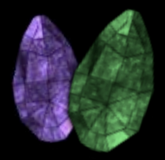
BAD_GUYS

Contact Callback

Player1

Player2

GOOD_GUYS



Collectable Power Ups

POWER_UPS

Collision Groups

Define logical groups

```
const uint32_t GOOD_GUYS    = 0x1 << 0;
const uint32_t BAD_GUYS     = 0x1 << 1;
const uint32_t POWER_UPS    = 0x1 << 2;
```

Baddies

BAD_GUYS

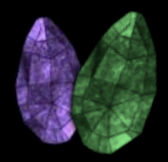
Contact Callback

Player1

Player2

GOOD_GUYS

Contact Callback



Collectable Power Ups

POWER_UPS

Collision Groups

```
const uint32_t GOOD_GUYS   = 0x1 << 0;
const uint32_t BAD_GUYS    = 0x1 << 1;
const uint32_t POWER_UPS   = 0x1 << 2;

/* players collide with bad guys, but not each other */

player1.physicsBody.categoryBitMask    = GOOD_GUYS;
player1.physicsBody.collisionBitMask   = BAD_GUYS;
player1.physicsBody.contactTestBitMask = BAD_GUYS | POWER_UPS;

player2.physicsBody.categoryBitMask    = GOOD_GUYS;
player2.physicsBody.collisionBitMask   = BAD_GUYS;
player2.physicsBody.contactTestBitMask = BAD_GUYS | POWER_UPS;


/* bad guys collide with players and other bad guys */

for (SKSpriteNode *badGuy in badGuys) {
   badGuy.physicsBody.categoryBitMask    = BAD_GUYS;
   badGuy.physicsBody.collisionBitMask   = BAD_GUYS | GOOD_GUYS;
   badGuy.physicsBody.contactTestBitMask = GOOD_GUYS;
}
```

Additional Sprite Kit Features

Sprite Kit Features

- SKScene transitions
- Reversing actions
- SKView debugging stats
- Automatic texture atlas creation
- Applying CIFilters to SKTextures
- Developer documentation
  - Programming guide
  - Code Explained: Adventure

Apple Evangelists

Contact information

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

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

Developer Documentation
http://developer.apple.com/library/

Related Sessions

Integrating with Game Controllers	Pacific Heights Tuesday 3:15PM
Designing Games with Sprite Kit	Mission Wednesday 2:00PM

Labs

Sprite Kit Lab	Graphics and Games Lab Wednesday 3:15 PM
Sprite Kit Lab	Graphics and Games Lab Thursday 9:00 AM

WWDC2013