Extracted from:

Build iOS Games with Sprite Kit

Unleash Your Imagination in Two Dimensions

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Build iOS Games with Sprite Kit

Unleash Your Imagination in Two Dimensions

Jonathan Penn
and Josh Smith

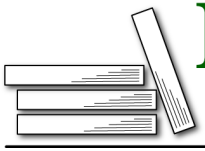
Edited by Rebecca Gulick

Build iOS Games with Sprite Kit

Unleash Your Imagination in Two Dimensions

Jonathan Penn
Josh Smith

The Pragmatic Bookshelf
Dallas, Texas • Raleigh, North Carolina



Pragmatic Bookshelf

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at *http://pragprog.com*.

The team that produced this book includes:

Rebecca Gulick (editor)
Potomac Indexing, LLC (indexer)
Cathleen Small (copyeditor)
David J Kelly (typesetter)
Janet Furlow (producer)
Ellie Callahan (support)

For international rights, please contact *rights@pragprog.com*.

Printed in the United States of America.
ISBN-13: 978-1-94122-210-2
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—July 2014

Playing Sound Effects in the Scene

When you're ready to implement one-shot sound effects in your game, there's no need to reach for the iOS audio APIs. Sprite Kit gives us everything we need with a special sound action. Let's add some shooting and explosion effects for the collisions.

First, we need to make sure the sound files are in the Xcode project. If you're creating the project yourself on the fly, drag the obstacleExplode.m4a, shipExplode.m4a, and shoot.m4a files into the Xcode file sidebar. Make sure the SpaceRun target checkbox is checked, like in Figure 7, *Dragging and dropping files into the Xcode project*, on page ?. Xcode doesn't always check that box for you for some file types.

Next, we need three properties on our scene object to hold the sound actions. Let's add these property definitions to the class extension of RCWMyScene.m.

02-Actions/step06/SpaceRun/RCWMyScene.m

```
@interface RCWMyScene ()
@property (nonatomic, weak) UITouch *shipTouch;
@property (nonatomic) NSTimeInterval lastUpdateTime;
@property (nonatomic) NSTimeInterval lastShotFireTime;
➤ @property (nonatomic, strong) SKAction *shootSound;
➤ @property (nonatomic, strong) SKAction *shipExplodeSound;
➤ @property (nonatomic, strong) SKAction *obstacleExplodeSound;
@end
```

We're creating these properties because loading a sound file into an action takes a moment and can introduce a brief pause in the gameplay the first time the sound is loaded and cached. We want to preload the sounds ourselves to get the delay out of the way up front during initialization, and we'll keep a strong reference to the actions so they aren't purged from Sprite Kit's sound cache for any reason. We know we always need these sounds available for this scene.

With the properties in place, create the sound actions and assign them to the properties in the -initWithSize: method.

02-Actions/step06/SpaceRun/RCWMyScene.m

```
- (id)initWithSize:(CGSize)size
{
    if (self = [super initWithSize:size]) {
        self.backgroundColor = [SKColor blackColor];
        NSString *name = @"Spaceship.png";
        SKSpriteNode *ship = [SKSpriteNode spriteNodeWithImageNamed:name];
        ship.position = CGPointMake(size.width/2, size.height/2);
        ship.size = CGSizeMake(40, 40);
```

```
        ship.name = @"ship";
        [self addChild:ship];

        self.shootSound =
            [SKAction playSoundFileNamed:@"shoot.m4a" waitForCompletion:NO];
        self.obstacleExplodeSound =
            [SKAction playSoundFileNamed:@"obstacleExplode.m4a"
                      waitForCompletion:NO];
        self.shipExplodeSound =
            [SKAction playSoundFileNamed:@"shipExplode.m4a" waitForCompletion:NO];
    }
    return self;
}
```

The -playSoundFileName:waitForCompletion: method takes a name for a sound file in the bundle for the first parameter. This can be any file that the iOS sound APIs understand, such as MP3, M4A, AIF, CAF, WAV, and more. While there are performance considerations when choosing sound-file types for the low-level iOS APIs, we don't have to worry about that for short one-shot sounds. M4A files are small, high quality, and quite sufficient for our use.

Notice how we set the waitForCompletion: parameter to NO for all these actions. This controls how the action fits in with the rest of the actions playing on the node. In this particular case, it doesn't really matter because we're just playing the sounds by themselves. But if they were part of an action sequence, then setting waitForCompletion: to YES would pause the sequence until the sound file stopped playing. That can be useful if you are chaining sounds together one after the other for effect. But for self-contained sound actions like this, we set the parameter to NO so that Sprite Kit knows we just want to trigger the sound and move on immediately.

With the sound actions initialized and ready to go, we'll add two lines to our -checkCollisions method to run the sound actions on the scene when either the ship or an obstacle explodes.

02-Actions/step06/SpaceRun/RCWMyScene.m

```
[self
 enumerateChildNodesWithName:@"obstacle"
 usingBlock:^(SKNode *obstacle, BOOL *stop) {
    if ([ship intersectsNode:obstacle]) {
        self.shipTouch = nil;
        [ship removeFromParent];
        [obstacle removeFromParent];
        [self runAction:self.shipExplodeSound];
    }
    [self
     enumerateChildNodesWithName:@"photon"
     usingBlock:^(SKNode *photon, BOOL *stop) {
```

```
        if ([photon intersectsNode:obstacle]) {
            [photon removeFromParent];
            [obstacle removeFromParent];
➤           [self runAction:self.obstacleExplodeSound];
            *stop = YES;
        }
    }];
}];
```

We're passing the sound action we want to play to the -runAction: method on the scene. That's all it takes to play a sound!

Notice that we're playing the sounds on the scene itself, and not on the other nodes. That's because we're removing those nodes from the scene, and any node that doesn't belong on an active scene doesn't run its actions.

Let's do the same thing to play a sound every time we shoot our photon torpedoes.

02-Actions/step06/SpaceRun/RCWMyScene.m

```
- (void)shoot
{
    SKNode *ship = [self childNodeWithName:@"ship"];

    SKSpriteNode *photon = [SKSpriteNode spriteNodeWithImageNamed:@"photon"];
    photon.name = @"photon";
    photon.position = ship.position;

    [self addChild:photon];
    SKAction *fly = [SKAction moveByX:0
                                    y:self.size.height+photon.size.height
                             duration:0.5];
    SKAction *remove = [SKAction removeFromParent];
    SKAction *fireAndRemove = [SKAction sequence:@[fly, remove]];
    [photon runAction:fireAndRemove];

➤   [self runAction:self.shootSound];
}
```

We're passing the sound action stored in the self.shootSound property to the -runAction: method on the scene.

Playing simple one-shot sounds is so easy with Sprite Kit. You can find out more about supported file formats in Apple's documentation.[4]

You're almost finished learning about node actions. For our last trick, we'll implement power-ups for the player's weapon.

4. https://developer.apple.com/library/ios/documentation/AudioVideo/Conceptual/MultimediaPG/UsingAudio/UsingAudio.html