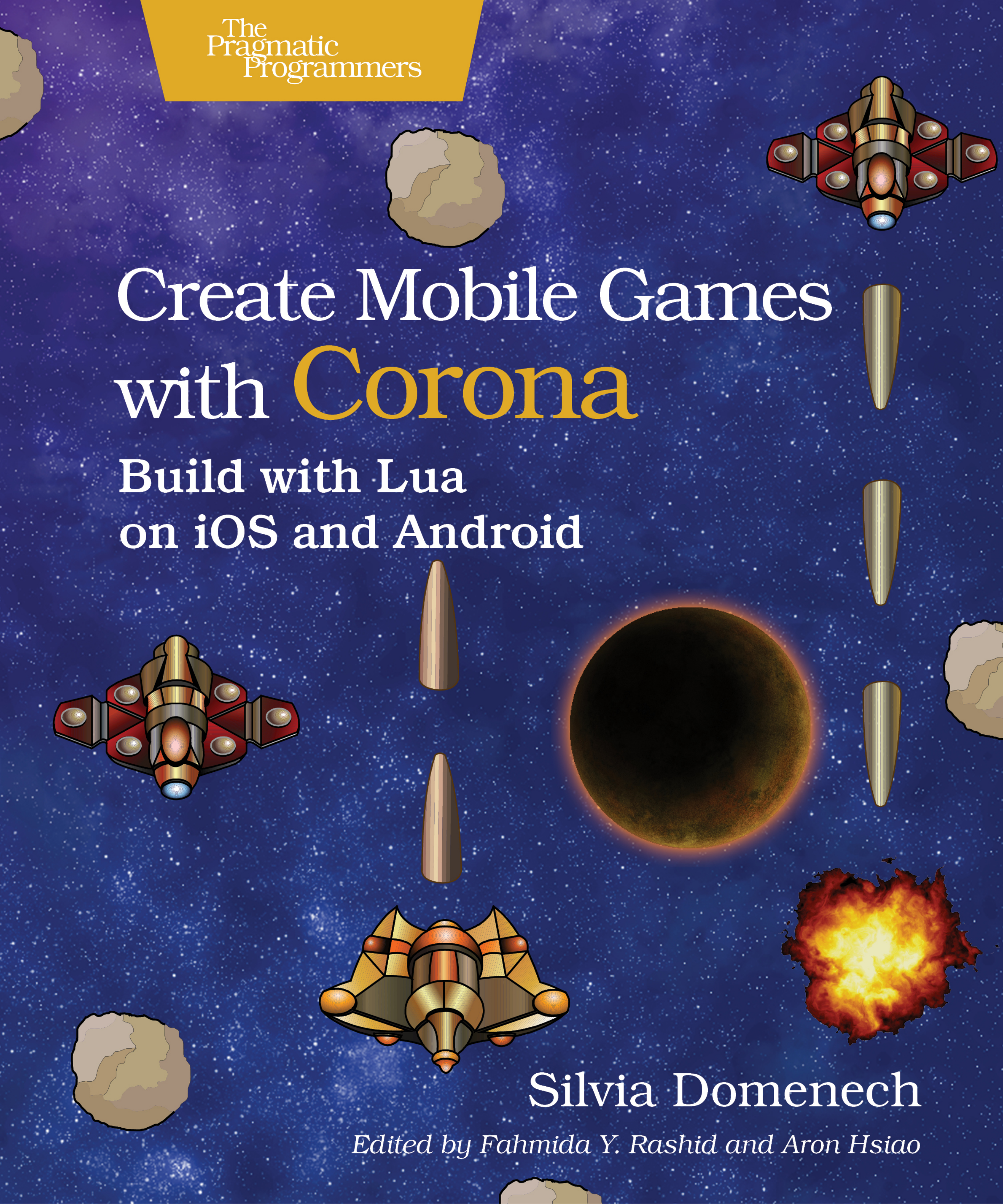
Create Mobile Games with Corona

Build with Lua on iOS and Android

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers



Create Mobile Games with Corona

Build with Lua on iOS and Android

Silvia Domenech

Edited by Fahmida Y. Rashid and Aron Hsiao

Create Mobile Games with Corona

Build with Lua on iOS and Android

Silvia Domenech

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

Pragmatic Bookshelf

The team that produced this book includes:

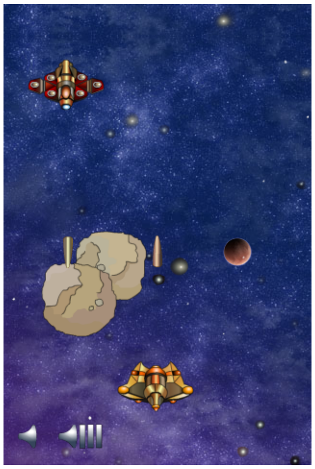
Fahmida Y. Rashid and Aron Hsiao (editors)
Potomac Indexing, LLC (indexer)
Candace Cunningham (copyeditor)
David J Kelly (typesetter)
Janet Furlow (producer)
Juliet Benda (rights)
Ellie Callahan (support)

Printed in the United States of America.
ISBN-13: 978-1-937785-57-4
Encoded using the finest acid-free high-entropy binary digits.
Book version: P2.0—January 2014

Galactic Warfare is relatively complete but lacks sound, and games without sound are boring. While you're completely right that there are no sounds in space, players generally expect to hear explosions and sound effects in space games. Since we don't want them to feel like Galactic Warfare is missing a fundamental component, we'll add sounds to the game. If the players want to be realists, they can choose to disable the sounds.

7.1 What You'll Learn

In this chapter, we'll do the following:



- Learn about the Corona Sound application programming interface (API)
- Add sounds to the game
- Create playlists for background music tracks
- Add volume controls and mute buttons

It's not enough to just add sound effects. It's as important to add them well. Otherwise, we'll be confronted with hordes of angry Galactic Warfare players we've irritated with wrong sound choices, loud sounds, or volumes that can't be changed. A bad review like "One star, needs a mute button!" is a silly way to hurt your app's ratings (not to mention sales) in the app stores. Once we add these changes, our app will look like the image shown earlier.

Since we'll create a few interface elements in this chapter, you'll need to be familiar with the graphics and interface methods we've covered in previous chapters. In particular, before starting this chapter, you might want to give Chapter 4, *Input and Menus*, on page ?, another look as a refresher on buttons and touch event listeners.

7.2 Finding Game Sounds

Though adding sound does involve programming, we'll first need to find, license, or create music and effects that match the atmosphere and style of our game. This isn't a secondary task; a bad soundtrack or bad sound effects can spoil the entire gameplay experience for our audience.

For sound effects, we'll need sounds of the right tone and length that also complement one another, and we'll need to coordinate their volume levels to avoid inadvertently highlighting some at the expense of others. Explosions shouldn't last for twenty seconds, for example, and bullets ought to make less noise than bombs. It's similarly unhelpful to add very quiet or muffled sound effects to a game, since players may not notice them in the first place.

Similar considerations apply to background music tracks. For a relaxed game, beat-heavy hip-hop music isn't the best choice. Though several music styles might be appropriate within a single game, if they play one after another in sequence, they should complement each other when heard in any order.

It's not uncommon for different sound files to need work before they'll fit together nicely in a game. The easiest solution when sounds need work is to use an audio-editing program to improve them, editing them as necessary and equalizing their volumes appropriately. Fixing sounds in game code by setting different volumes for each sound effect or other similar tricks gets too confusing to be practical, particularly when lots of sounds are involved.

If you prefer to download or buy sounds for your game rather than hire a composer, look to stock sound websites, which are both cheap and effective. For Galactic Warfare we'll use sound effects from SoundBible and background music loops from Incompetech (the game code includes a complete list of our sounds). Soundsnap, AudioJungle, and Soundrangers are also fantastic places to find music with a variety of licenses and prices, while Freesound and freeSFX both offer free sound effects for very tight game-development budgets. Appendix 1, *Corona Resources*, on page ?, lists a few additional resources.

7.3 Understanding the Corona Sound API

Corona, like Flash and other rapid application-development platforms, uses channels to play sounds. Every active channel is a sound or tune that the user can hear. These channels play independently of one another, and we can play a different tune or sound in each of Corona's thirty-two channels. To add sound effects and background tracks to Galactic Warfare, we'll have to learn how to find an unused channel, start and stop sound playback, and remove completed sounds.

Most of the time, we load sounds using Corona's audio.loadSound() function, which returns an *audio handle*, something like a pointer to the sound in question. After loading sounds, pass their audio handles to audio.play() to play sounds back. In that sense, handles are similar to the sound tools in other development environments and are easy to use in the simplest case, shown here:

SoundPlay/main.lua

```
-- Load and play mysound.wav
mySound = audio.loadSound( "mysound.wav" )
```

```
myChannel = audio.play( mySound )
```

Each time we play a sound, Corona assigns the audio to an unused channel ranging from 1 to 32, but we can also tell Corona which channel to use if we want to do so. As long as we know what channel is being used for a sound's playback, we can mute it or change its volume. In Galactic Warfare we'll play all of our background music through one channel (making sure we remember which one it is) and let Corona manage the channels for the rest of our sounds.

7.4 Adding Sound Effects

Now that we know where to find sounds, which sounds to choose, and the basics of sound playback in Corona, we can add sound to Galactic Warfare. Let's begin by adding effects for our shots and explosions so that they can be heard as well as seen, expanding just a bit on the basic sound-playback technique shown earlier.

Loading and Playing Sounds

The basic audio.loadSound() and audio.play() methods shown earlier are fine for a small app with one or two sounds, but what about a big game with dozens of sounds? We could add each sound manually, but that would be a lot more work and involve lots of repeated code, making changes more difficult down the road.

For this reason, we'll use a table to store sound filenames and properties instead and then loop through the table to load each sound and add its audio handle to a table of audio handles.

Begin by listing the audio files for our game in globals.lua using the SOUND_EFFECTS variable. We'll stick to sound effects for now, but we'll do something similar later for background music. The two sound effects we'll use are called explosion.wav and missile.wav and can be found in the chapter's code files.

Sound/globals.lua
```
SOUND_EFFECTS = {
    SOUND_EXPLOSION = "sounds/explosion.wav",
    SOUND_MISSILE = "sounds/missile.wav"
}
```

In addition to the variable listing our sound files, we need a variable to store the sound handles that result from loading them. Using a table will let us access all of our game's sound-effect handles under a single name. First define this variable, calling it something like soundEffectHandles.

Sound/game.lua
```
local soundEffectHandles
```

Now that we have one table listing the files we'll need for sound effects and another one ready to store their handles, we're ready to step through the first table to load the sound effects using audio.loadSound(). Store the handles for each sound in the variable we just defined (soundEffectHandles), and assign names to the table rows so that we can access them by name. Since we're working with explosion and missile sounds, call them explosion and missile.

Sound/game.lua
```
soundEffectHandles = {
    explosion = audio.loadSound( SOUND_EFFECTS.SOUND_EXPLOSION ),
    missile = audio.loadSound( SOUND_EFFECTS.SOUND_MISSILE )
}
```

With our sounds ready to go, let's edit the Galactic Warfare game loop to play a sound effect each time a missile or an explosion is added to the game. We added explosion animations in the updateEnemies() function in game.lua, so we can now call the audio.play() function right after adding the explosion sprite.

Sound/game.lua
```
local chan = audio.play( soundEffectHandles.explosion )
```

We can repeat the same steps for the shooting sound. We added bullets using the Bullet() constructor, but the tricky part is that both the player and the enemies can shoot, so we'll have to add bullet sounds twice. We'll start by adding sounds for player bullets, which are created in the game's tick() function. We'll add the audio.play() call there, right after adding each player bullet to the game.

Sound/game.lua
```
local chan = audio.play( soundEffectHandles.missile )
```

We called the enemy update() functions from updateEnemies(), so we'll make our enemy sound changes to game.lua. Add the same audio call right after we add a new enemy bullet to the bullets table. We've used the same sound effect for both player and enemy bullets to keep things simple, so we can use the same code as before.

Sound/game.lua
```
local chan = audio.play( soundEffectHandles.missile )
```