

208: Scene Kit

Part 4: Challenge Instructions

Copyright © 2014 Razeware LLC.

All rights reserved. No part of this book or corresponding materials (such as text, images, or source code) may be reproduced or distributed by any means without prior written per- mission of the copyright owner.

This book and all corresponding materials (such as source code) are provided on an "as is" basis, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in action of contract, tort or otherwise, arising from, out of or in connec- tion with the software or the use or other dealings in the software.

All trademarks and registered trademarks appearing in this book are the property of their respective owners.

# Road kill

Your frogger-clone is starting to shape up. But there is one thing missing before this game is any fun: A challenge.

Your challenge will be to add cars to the roads based on what you have learned so far:

* 1. **Add nodes to the scene** for the cars
  2. **Load geometry** to make the cars actually look like cars
  3. **Put a material** on the car geometry to make the car look nice.
  4. **Make the car drive** along the road using actions.

Once you have finished this challenge, you will have made a bare-bone frogger-clone with all the basic mechanics.

Start by opening the Xcode project in the **Lab** folder of the tutorial materials.

## Challenge A: Nodes and geometry

The challenge project already includes a COLLADA file for the cars called **car.dae** in the **Models** folder of **assets.scnassets**.

Your challenge is to create a method with the following signature in **GameScene.swift**:

func createCarAtPosition(#position: SCNVector3, flipped: Bool) -> SCNNode

You then need to add the code to load the **car.dae** scene, extract the correct node.

**Hint:** Use the SceneKit editor to do a look-up of the correct name for the node that contains the geometry for the car.

Then you need to return the created node.

The challenge project contains another method called setupCarSpawnNodes() that will actually create the cars. You do not have to worry about this method for now but just know that it is there.

## Challenge B: Materials

Did you get cars onto the roads? Very good! But they look a bit dull, right? Your next challenge is to change that by adding a material to the car.

You’ll extend the createCarAtPosition(position:) in this challenge by adding a material that uses the **texture model\_texture.tga** in the **Models** folder in **assets.scnassets**.

**Hint:** The texture should be the contents of the Diffuse materialProperty.

When you have added the code to apply the material, do a build and run and look at the draw count. That’s A LOT of draw calls :] Why are there so many draw calls now? You will get the answer in the conclusion of this tutorial :]

## Challenge C: Actions.

If you completed challenge B you now have some very nice, but static, cars on your scene. Your last challenge will be to make them move.

To make the cars move, you have to add some code to the setCarSpawnNodes() method. To make it more interesting, you need to make the cars in the odd rows move to the left and the cars in even rows move right (x-axis).

**Hint:** Use the moveDirection variable to determine if a row is even or odd.

The setCarSpawnNodes() method already contains a lot of code. What it does is determine what rows in the level are roads. Depending on whether the row is odd or even, the node is positioned either to the left or right of the road.

The code you need to add should be inserted after the comment in the setCarSpawnNodes() method:

// TODO: Add the code to make the car move’

**Hint:** The helper class **GameLevel.swift** contains a method called gameLevelWidth() that returns a float that will let you know the total width of the level. You can use this to tell how far the car needs to move to move out of the screen.

Last part of this challenge is to ensure that the car is removed from the scene once it moves out of the level.

**Hint:** This involved adding a sequenced action with the action you created for the movement and adding a removeFromParent() action to the end.

Perfect, you are now able to make your own games in Scene Kit :]

## Solutions

There are many ways you could have solved these challenges and there are no right or wrong ways. Of cause, some solutions are more performant than others but in general the most important thing is not to optimize until there is a performance issue – that is the golden rule of making games :]

If you want to see how the instructor solved these challenges you’ll find am Xcode project in the **Challenge** folder of the tutorial materials.